POLITECNICO DI TORINO
Repository ISTITUZIONALE

Towards an Automatic Framework for Solving Optimization Problems with Quantum Computers

Original
Towards an Automatic Framework for Solving Optimization Problems with Quantum Computers / Volpe, D., Quetschlich, N., Graziano, M., Turvani, G., Wille, R.. - ELETTRONICO. - 473:(2024), pp. 46-57. (IEEE International Conference on Quantum Software (QSW) Shenzhen (China) 07-13 July 2024) [10.1109/qsw62656.2024.00019].

Availability:
This version is available at: 11583/2992492 since: 2024-09-23T06:27:06Z

Publisher:
IEEE

Published
DOI:10.1109/qsw62656.2024.00019

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

18 November 2026

A Predictive Approach for Selecting the Best Quantum Solver for an Optimization Problem

Deborah Volpe*, Nils Quetschlich†, Mariagrazia Graziano‡, Giovanna Turvani*, and Robert Wille†§
*Department of Electronics and Telecommunications, Politecnico di Torino Italy
†Chair for Design Automation, Technical University of Munich, Germany
‡Department of Applied Science and Technology, Politecnico di Torino Italy
§Software Competence Center Hagenberg GmbH (SCCH), Austria
deborah.volpe@polito.it, nils.quetschlich@tum.de, mariagrazia.graziano@polito.it,
giovanna.turvani@polito.it and robert.wille@tum.de

***Abstract*—Leveraging quantum computers for optimization problems holds promise across various application domains. Nevertheless, utilizing respective quantum computing solvers requires describing the optimization problem according to the *Quadratic Unconstrained Binary Optimization* (QUBO) formalism and selecting a proper solver for the application of interest with a reasonable setting. Both demand significant proficiency in quantum computing, QUBO formulation, and quantum solvers, a background that usually cannot be assumed by end users who are domain experts rather than quantum computing specialists. While tools aid in QUBO formulations, support for selecting the best-solving approach remains absent. This becomes even more challenging because selecting the best solver for a problem heavily depends on the problem itself.**

In this work, we are accepting this challenge and propose a predictive selection approach, which aids end users in this task. To this end, the solver selection task is first formulated as a classification task that is suitable to be solved by supervised machine learning. Based on that, we then propose strategies for adjusting solver parameters based on problem size and characteristics.

Experimental evaluations, considering more than 500 different QUBO problems, confirm the benefits of the proposed solution. In fact, we show that in more than 70% of the cases, the best solver is selected, and in about 90% of the problems, a solver in the top two, i.e., the best or its closest suboptimum, is selected.

This exploration proves the potential of machine learning in quantum solver selection and lays the foundations for its automation, broadening access to quantum optimization for a wider range of users.

The pre-trained classifier is integrated into the MQT Quantum Auto Optimizer (MQT QAO) framework, publicly available on GitHub (https://github.com/cda-tum/mqt-qao) as part of the Munich Quantum Toolkit (MQT).

***Index Terms*—Machine Learning, Predictive Models, QUBO, Quantum Optimization, Quantum Annealer, Quantum Approximate Optimization Algorithm, Variational Quantum Eigensolver, Grover Adaptive Search**

I. Introduction

Quantum computing holds promise for solving optimization problems across various application fields, including finance [1], resource allocation [2], and scheduling [3]. However, harnessing *quantum optimization*—i.e., solving optimization problems leveraging quantum computers—demands *expertise* in quantum computing, especially when formulating optimization problems in a format suitable for quantum computers such as QUBO and selecting the most promising solver to determine the result.

This poses a challenge for researchers and industries not directly involved in quantum computation, hindering their initial exploration of quantum solutions for real-world applications.

While some frameworks for supporting the translation of optimization problems into a quantum-compliant format exist, such as that proposed in [4]–[9], the literature lacks tools to assist in leveraging *quantum solvers*, including both selecting the *most promising solver* and its respective *parameters*. Unfortunately, executing and evaluating all quantum solvers and their parameters configurations can be expensive both in terms of time and money, especially considering the limitations of nowaday quantum computers, making this naive approach unfeasible. For these reasons, *automating* solver selection and its parameter configuration can significantly streamline the process, enabling users to achieve good results with minimal effort and expanding access to quantum optimization for a broader range of users.

This work proposes approaching the challenge of selecting solvers as a *classification* task, considering the most popular quantum solvers—i.e., *Quantum Annealer*, *Quantum Approximate Optimization Algorithm*, *Variational Quantum Eigensolver*, and *Grover Adaptive Search*—and *supervised machine learning models*. In addition to quantum solvers, the model also considers the classical *Simulated Annealing*, the classical counterpart of Quantum Annealing. By including a classical solver, we demonstrate the versatility of the approach and that the model could be further expanded by adding more classical and quantum solvers. In this way, the classifier can identify the optimal quantum solver and distinguish problems that could gain advantages from quantum exploration compared to classical methods [10]. Moreover, in this article, some techniques for *adjusting solver parameters* based on problem size and characteristics are proposed.

The considered dataset has been obtained by solving more than *500* different QUBO problems. The obtained results are promising since the solvers' setting guarantees, on average, promising results, and the best solver is identified in *more than 70%* of the cases. Moreover, a solver in *the top two*, i.e., the best or its closest suboptimum, is selected in *about 90%* of the considered problems.

This exploration proves the potential of supervised machine learning in selecting the best solver for a problem, significantly enhancing the accessibility of quantum optimization across a

wider range of users. The predictor streamlines the solver selection process, hiding its complexity into a black box solution. Consequently, end users can effortlessly evaluate the quantum solution's effectiveness for their application without deep quantum solver expertise. Moreover, this approach mitigates the resource-intensive task of executing multiple solvers in terms of time and cost, providing a more efficient pathway to identify potential quantum advantages. The pre-trained classifier is integrated into the MQT Quantum Auto Optimizer (MQT QAO) framework, publicly available on GitHub (https://github.com/cda-tum/mqt-qao) as part of the Munich Quantum Toolkit (MQT, [11]).

The rest of the article is organized as follows. Section II briefly reviews quantum optimization, focusing on the workflow needed for solving an optimization problem with quantum computers and quantum solvers. Section III outlines the motivation behind this work and discusses the general idea. The proposed supervised learning approach for solver selection and the considered solver setting are presented in Section IV, focusing on the methodology adopted for the classification model development. The obtained results are provided in Section V, while the effectiveness of the proposed approach is examined in Section VI. Finally, in Section VII, conclusions are drawn, and future perspectives are illustrated.

II. Solving Optimization Problems with Quantum Computers

This section reviews quantum optimization, concentrating on the quantum-compliant problem formulation and quantum solvers. Moreover, the main steps required for solving generic optimization problems with quantum computers are summarized.

To this end, two different paradigms of quantum computing exploited to solve optimization problems are considered:

- quantum annealing, implemented through a *special-purpose quantum computer* called *quantum annealer* [12]–[16];
- *quantum circuit model* [17], defining proper algorithms involving both classical and quantum or only *quantum computers general-purpose*.

However, independently from the exploited paradigm, the problem must be written into a compliant formulation involving only binary variables.

A. QUBO Model

The usual formulation for solving an optimization problem leveraging quantum computers is the *Quadratic Unconstrained Binary Optimization* (QUBO) model [18], [19]. This formulation describes the problem involving only binary variables. The *quadratic* term refers to the highest power applied to these binary variables, and *unconstrained* indicates that the constraints cannot be considered conventionally, but only via introduction of *quadratic penalties to the objective function*. Therefore, the corresponding objective function can be written as

$$f(\mathbf{x}) = c + \sum_i x_i \cdot a_i + \sum_{i<j} b_{ij} \cdot x_i x_j \,, \tag{1}$$

where $x_i \in [0, 1]$ is a binary variable, $x_i x_j$ is a coupler of two variables, a_i is a single-variable weight, b_{ij} is a strength that controls the influence of variables i and j, and c is an offset, which can be neglected during the optimization since it shifts the entire objective function without altering the extremal points.

B. Quantum Solvers

Different quantum solvers have been proposed for optimization problems provided according to the QUBO model. The most popular are briefly discussed in the following.

1) Quantum Annealer (QA, [12], [15], [16]) is a special-purpose quantum computer designed to solve optimization problems with the quantum annealing algorithm by harnessing *quantum superposition* and *tunneling* to explore the solution space of a given optimization problem.
It implements the quantum counterpart of a *Simulated Annealing* (SA, [20]) algorithm, which solves optimization problems by encoding them in the energy of a system whose thermal evolution for reaching the ground state is emulated to obtain the optimal solution. Analogously, in QA, the quantum system begins in a superposition of states and evolves following the principles of the quantum adiabatic theorem towards its lowest energy state, i.e., the problem's optimal solution. The system evolution, shown in Figure 1, is managed by a schedule parameter known as the *annealing time* or *annealing schedule*, which determines the rate at which the system transitions from the initial to the final state. Properly setting this parameter is crucial to ensure *adiabaticity*, which entails maintaining the system's ground state throughout the evolution process and achieving high-quality solutions.
Quantum annealers comprise a set of qubits interconnected via tunable couplers, forming a network known as the *annealing graph*. The connectivity of this graph is constrained by the qubit technology. A *minor embedding* [21] step is employed to address this limitation and allows the mapping of a fully-connected problem onto a partially-connected device. It maps each QUBO variable to a set of strictly correlated physical qubits, overcoming the connectivity constraint. This implies that the number of qubits required for solving an optimization problem could be significantly higher than the binary variables involved.

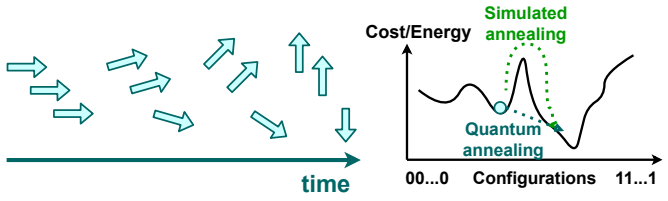


Fig. 1: Evolution of the Quantum Annealer system and comparison against Simulated Annealing exploration mechanism.

2) Quantum Approximate Optimization Algorithm (QAOA, [22], [23]) is a hybrid quantum-classical technique for solving optimization problems. It is a *variational algorithm*, i.e., an iterative procedure applied to some parameters of the quantum circuit, such as the rotation amounts of single-qubit gates.

These parameters are optimized classically such that the final quantum state encodes the optimal solution of the problem. The variational circuit provides an overall unitary evolution composed of a layer depending on problem Hamiltonian and a mixed state Hamiltonian, as shown in Figure 2. This is chosen to mimic quantum adiabatic evolution behavior through Trotterization, a technique for approximating the continued evolution with discretized steps. This is done by decomposing the evolution operator into a series of simpler time-independent operators by using the Trotter-Suzuki method [24]. An initialization state must be obtained with a state preparation circuit to guarantee a proper evolution. Therefore, the key parameters of the algorithm are the number of steps for Trotter discretization, called repetitions (reps), the state preparation circuit, the mixed state, and the classical optimizer. This solver demands the execution of circuits with a number of qubits equal to the number of QUBO variables.

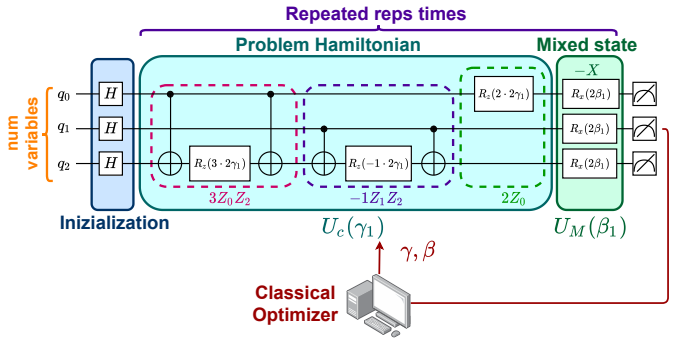


Fig. 2: Quantum Approximate Optimization Algorithm.

3) Variational Quantum Eigensolver (VQE, [25], [26]) is a hybrid quantum-classical algorithm that aims to minimize a scalar cost function mapped onto a Hamiltonian. It is a variational algorithm in which the quantum circuit, as shown in Figure 3, is composed of an ansatz whose parameters are properly optimized by a classical solver such that the expectation value of the system with respect to the Hamiltonian of the problem is minimized. Therefore, the key parameters of the algorithms are the ansatz and the classical optimizer. This solver requires the execution of circuits with a number of qubits equal to the number of QUBO variables.

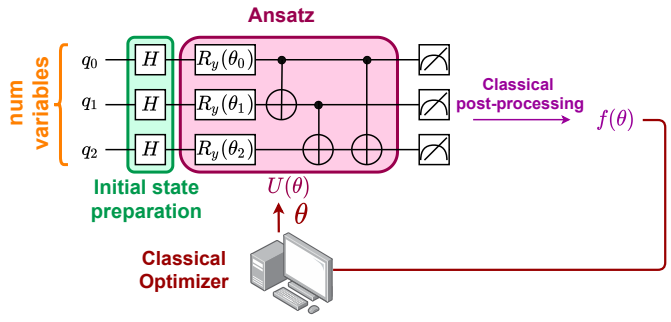


Fig. 3: Variational Quantum Eigensolver.

4) Grover Adaptive Search (GAS, [27]–[31]) is a hybrid quantum-classical algorithm that leverages a successive approximation approach to minimize a cost function. In particular, the algorithm works by repeatedly sampling negative values of the objective function and then incrementally shifting it upward by the same amount until no further negative values are observed. Therefore, the last negative sample represents the minimum of the objective function. Negative samples are obtained through the *Grover Search* routine as shown in Figure 4, where the cost function is encoded onto the quantum state using a quantum dictionary, which allows the expression of a key (function domain) and value (function image) relation. Consequently, the exploited quantum circuit requires a number of qubits equal to $n_{\text{keys}}+n_{\text{values}}$, where n_{keys} is the number of binary variables and n_{values} is the number of bits necessary for representing the image of the function. Selecting the appropriate n_{values} poses an initial challenge in utilizing this algorithm since it necessitates knowledge of the function's operational range.
Since the optimal number of Grover rotations r depends on the number of negative samples over the entire solution space that cannot be predetermined, techniques have been proposed in [28], [31], [32] for choosing it in each algorithm step. Another critical aspect of the algorithm involves determining whether negative values remain. Although obtaining a positive sample via Grover search is an event that occurs when non-negative function values are available, this cannot serve as a direct terminating condition, as it may also occur when the number of Grover rotations is improperly set. Consequently, the standard approach is to count the number of consecutive measured positive samples and stop the algorithm when this number overcomes a certain threshold th to be properly chosen.

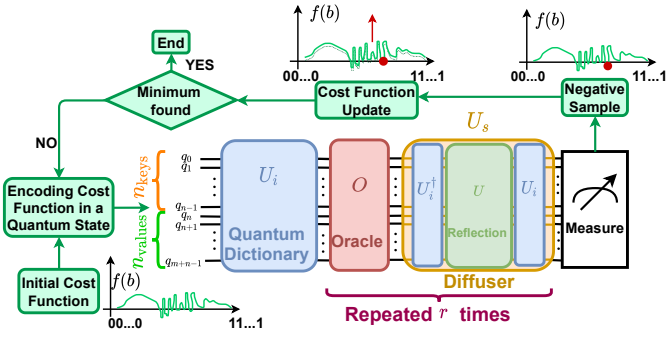


Fig. 4: Grover Adaptive Search

C. From Optimization Problems to Quantum Solutions

Using the basis and methods revised above, the workflow shown in Figure 5 emerged, enabling end users to solve optimization problems leveraging quantum computers or quantum annealers. More precisely the steps needed for harnessing quantum solvers are discussed in the following.

1) Problem specification: The initial step of the optimization process involves defining the problem specifications, which include the *variables*, *objective functions* (criteria for optimization), and any *constraints* required for a valid solution.

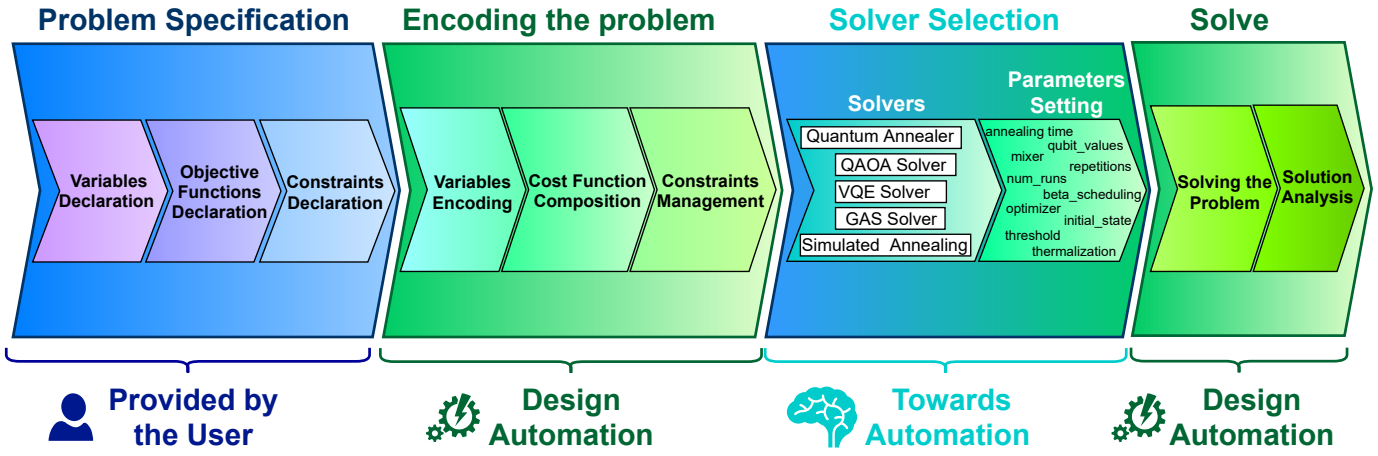


Fig. 5: Quantum optimization flow.

These variables can take on *binary*, *discrete*, or *continuous* values, delineating their characteristics such as unipolar/bipolar nature or defined value ranges. The objective function represents a parametric description of a figure of merit, with its optimal value either being the minimum or maximum, necessitating the specification of the optimization direction.

2) Encoding the Problem: Afterwards, the problem has to be described in a *solver-compliant* form, particularly employing the QUBO model. As mentioned, this model exclusively supports *binary variables*, requiring proper *encoding mechanisms* for expressing continuous and discrete ones. In the case of multi-objective optimization, the aggregation approach can be employed to merge objective functions into a higher scalar function, which reflects a preference criterion, *composing the cost function*. Moreover, *constraints* can be managed only as *weighted penalty functions* incorporated within the primary objective function. Finally, any eventual higher-order polynomials need to be reduced to second-order terms.

3) Solver Selection: Following the problem specification stage, the next step involves selecting a solver, such as QA, QAOA, VQE, or GAS. Each of these optimizers has a peculiar exploration mechanism. Therefore, different performance is expected when considering a solver instead of another. Subsequently, careful attention must be paid to selecting suitable settings for the chosen solver. For example, in the case of GAS, a proper number of qubits for the value and the threshold has to be selected.

4) Solve: The resulting QUBO formulation is then submitted to the solver with an *appropriate parameter configuration* (Solving the Problem). Due to their stochastic nature, solvers are typically executed multiple times, and the best result is considered. Access to quantum devices is possible via *cloud services*, through an account. Otherwise, quantum solvers such as QAOA, VQE, and GAS can also be executed by exploiting *classical simulators*, as those presented in [33]–[38].

Finally, the acquired solution must be decoded to retrieve the original problem variables; then, its quality must be assessed (Solution analysis). This entails evaluating the initial cost functions using the obtained configuration and verifying constraint satisfaction.

D. Existing Tools

These steps are rather complex and can easily overwhelm users who are experts in their respective domains (such as finance [1], resource allocation [2], and scheduling [3]) rather than in quantum computing. Therefore, this procedure may benefit from design automation to enhance the accessibility of quantum solutions.

In recent years, several libraries and some tools have emerged to streamline the QUBO formulation. The main libraries include *pyqubo* [39], [40], *qubovert* [41], *dimod* [42], *Qiskit-optimization* [43], *Fixstarts Amplify* [44] and *open-QAOA Entropica* [45]. They significantly facilitate complex aspects of the QUBO formulation process. Nonetheless, their primary limitation lies in their lack of support for fully automating these steps, thereby restricting their accessibility to users with at least a minimum level of expertise in the field. Moreover, in response to user demands for comprehensive automation, three automation frameworks have surfaced in the last two years: *AutoQUBO* [4]–[6], QUBO.jl [7], and MQT QAO [8]. Their focus primarily lies in assisting with the problem encoding stage and interface with solvers. Finally, tools and methods exist that guide end-users through the corresponding compilation process [46].

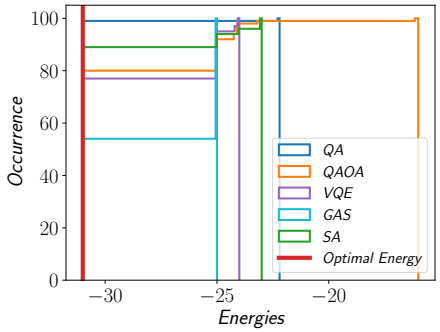
Nevertheless, to our knowledge, no tool is currently available in the literature to aid in the solver selection stage despite its not negligible complexity, especially for non-expert users, since it requires deep knowledge of quantum solvers, which may inhibit their interest in experimenting with quantum solutions. Addressing this gap remains an ongoing challenge that we aim to address in this manuscript.

III. Towards Automation of Quantum Solvers Selection

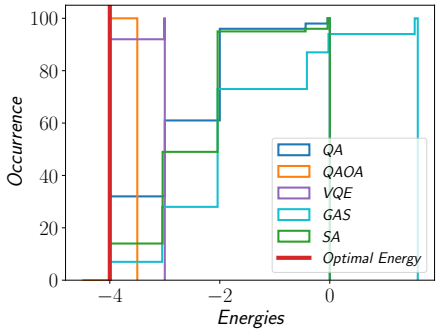
This section discussed the problem selection challenges, including both the choice of the optimization solver and the setting of its parameters, for end-users and the unmet needs that this work aims to address.

A. Considered Challenge

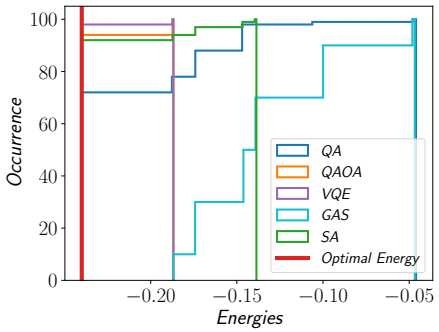
Estimating a priori what might be the *best solver for the problem of interest* or whether a classical or quantum approach might be better is a non-trivial task.



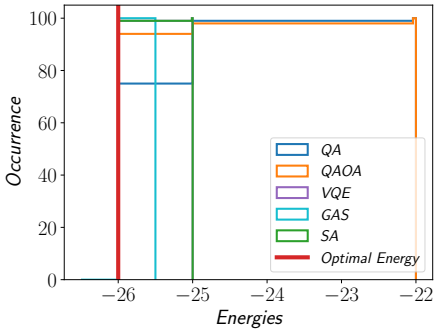
(a) Cumulative distribution of a Set Packing problem, where QA emerges as better solver



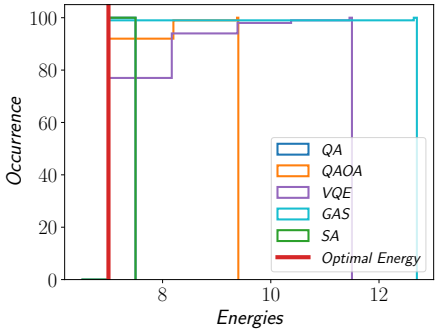
(b) Cumulative distribution of a Kclique problem, where QAOA emerges as better solver



(c) Cumulative distribution of a Portfolio Optimization problem, where VQE emerges as better solver



(d) Cumulative distribution of a Maxcut problem, where GAS emerges as better solver



(e) Cumulative distribution of a Minimum Vertex Cover problem, where SA emerges as better solver

Fig. 6: These figures show cumulative distributions obtained by solving different QUBO problems with QA, QAOA, VQE, GAS, and SA. To understand the meaning of these plots, one rule has to be considered: the probability of obtaining the optimal value (or a value close to it) with a specific solver is higher when its corresponding cumulative distribution is more concentrated on the left of the plot, where the lowest values of the objective function are located. It is possible to notice that the best solver, in terms of solution quality, varies across each problem examined. This observation highlights the dependence of solver effectiveness on the problem itself. To underline this argument, we have evaluated applications from various domains, leading to completely different results in terms of optimal solver for each particular case.

This problem is similar to the *No Free Lunch Theorem* [47], which declares that every optimization technique performs as well as every other one on average, and *a single best optimization algorithm does not exist*. Consequently, *different optimization problems correspond to different optimal solvers*, complicating a priori selection of the most suitable solver for a specific problem.

Example: *The problem is illustrated by the cumulative plots reported in Figure 6, considering five different optimization problems as examples. Cumulative plots are a common representation of heuristic solver results, showing the distribution of solver outcomes across multiple runs. To correctly interpret the plot, a simple rule has to be considered: the probability of obtaining the optimal value (or a value close to it) with a specific solver is higher when its corresponding cumulative distribution is more concentrated on the left of the plot, where the lowest values of the objective function are located. These examples show that each solver is the best for one problem, and a single best optimization approach does not exist.*

However, executing and evaluating all quantum solvers can be expensive in terms of time and money, particularly given the current limitations. Nowadays, running quantum circuit model solvers on simulators is necessary to get results not affected by errors introduced by hardware noise and non-ideality phenomena, such as qubit relaxation and decoherence. Moreover, accessing real quantum computers often entails subscription fees that cannot be neglected. These problems could inhibit the exploration of quantum solutions for real-world optimization scenarios.

Furthermore, properly *configuring quantum solver* parameters demands significant expertise in quantum computing, which is unusual among conventional optimization users. Also, in this case, assessing all possible parameters combinations to find the optimal configuration is prohibitively costly and impractical, especially given the current constraints on access to quantum resources and quantum simulators.

Moreover, adjusting scaling factors such as annealing time in QA or determining parameters like the number of qubits for

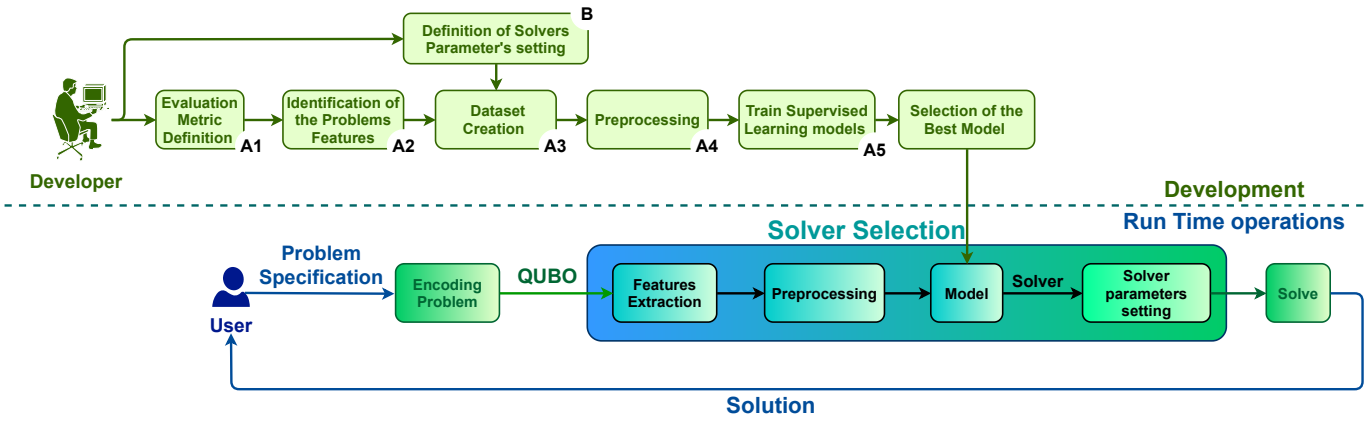


Fig. 7: Workflow for implementation of the proposed predictive model and for its exploitation from the user perspective. The letters and numbers correspond to the subsections where each step is discussed.

value representation in GAS could also be challenging even for users well-versed in quantum computing. Indeed, in the former case, it necessitates leveraging prior experience, while estimating the operational range of the function is required in the latter.

For all these reasons, the solver selection step can greatly benefit from *automation* for both choosing the best solver and setting its parameters according to the specific problem to solve.

B. General Idea

This work aims to offer tools for assisting non-expert users in *selecting quantum solvers and configuring their parameters* for any QUBO-representable optimization problem, hiding the complexities inherent in these tasks from the users.

For solver selection, we propose to leverage *supervised machine learning*. In order to develop this solution, the following steps are necessary:

1) Determining an *evaluation metric* for defining the best solver.
2) Extracting and identifying meaningful information from the QUBO formulation to serve as *features* characterizing the underlying optimization problem in the classification task.
3) Creating the *dataset*, which consists of:
   a) selecting a diverse set of benchmarks;
   b) defining a strategy for handling parity cases;
   c) analyzing the dataset's significance.
4) Applying data *preprocessing* techniques.
5) Evaluating performance with various *classification models* and selecting the most effective.
6) Finally, obtaining the trained models and exploiting them to predict the best solver for new problems runtime.

For configuring solver parameters, we propose leveraging experimental results from the literature to identify scaling trends and state-of-the-art techniques for extracting QUBO information from its formulation. For instance, to determine the number of repetitions in QAOA, we suggest using information/insights gained by plots available in the literature, like those in [48], [49], that depict the empirical scaling of this parameter in experimental results. This information can then be used to estimate a reasonable value based on the problem size. These parameters scaling can be further improved by executing additional tests.

IV. A Supervised Learning Approach for Solver Selection

Running all possible quantum solvers with even different parameters easily exceeds what is practically feasible due to the required effort to implement them and the associated costs for their execution. Therefore, in this work, we propose a methodology for *predicting the best solvers* among the available ones without explicitly executing all of them. In this article, we consider QA, QAOA, VQE, and GAS as quantum solvers and SA as a classical counterpart. In order to reach the target, the selection of the solver problem is interpreted as a classification task, addressable by exploiting *supervised machine learning techniques*. Furthermore, simple strategies for effectively configuring and scaling the parameters of quantum solvers are proposed.

From the users' perspective, the provided tool, integrated into the MQT Quantum Auto Optimizer framework that implements the problem translation, enables the achievement of good results without trying out all possible quantum solvers, even with different parameters, and any knowledge about quantum computing. Indeed, it can be seen as a black-box tool (the box in Figure 7) that automatically makes those decisions—in real-time and without any associated costs of actually executing any circuit and hiding the required steps from the users. The tool can then substitute the former Solver Selection step with the two tasks, as shown in Figure 5. However, realizing this black box requires significant effort on the developer's side. What this exactly means is explained in the following and shown in Figure 7.

A. Prediction of the Best Solver

As anticipated in Section III-B, several steps are required for developing a solver predictor. They are discussed in the following.

1) Evaluation Metric: First of all, establishing an *evaluation metric*, i.e., a singular score determining the best solver for each problem, is essential for developing the solver predictor. In general, an optimization problem solver is evaluated based on the *quality of the obtained solution* and the *time required to achieve convergence*. However, comparing convergence times among solvers in this context is non-trivial. Indeed, the currently available quantum circuit model devices do not allow reliable execution of quantum solvers, making for an adequate evaluation of their exploration quality mandatory execution on quantum simulators. Since simulators' complexity, and, consequently, execution time, grows exponentially with the number of involved qubits, a fair comparison with QA and SA is not permitted. For this reason, this study focuses only on the quality of the achieved solution, with execution time considerations left for future exploration when quantum computers' reliability improves within the quantum circuit model.

Given the stochastic nature of solvers, evaluating their outcomes requires *multiple executions* to assess solution quality comprehensively. In this work, we considered one hundred runs of each algorithm for every solver. We believed this number sufficient to gather meaningful statistical insights while avoiding excessive computational demands that could make the tests impractical without sophisticated high-performance computing platforms.

In order to consolidate various metrics related to the distribution of the overall results—i.e., the minimum value, the average and the variance of the obtained results, the percentage of optimal/suboptimal solutions, etc.—into a single figure of merit, we propose a scoring system composed of weighted indicators to minimize:

$$F_{\mathrm{s}} = -\alpha p_{\mathrm{s}} + \beta(E_{\mathrm{opt}} - E_{\mathrm{ref}}) + \gamma(E_{\mathrm{avg}} - E_{\mathrm{ref}}) + \delta E_{\mathrm{var}} - \eta p_{\mathrm{v}} \,, \quad (2)$$

where p_{s} represents the percentage of outcomes equal to the optimum, E_{opt} denotes the best-achieved value, E_{ref} stands for the reference optimal value for the problem, E_{avg} signifies the average value obtained, E_{var} indicates the variance of the obtained values, and p_{v} is the percentage of solutions satisfying the constraints. α, β, γ, δ, and η represent tunable weights. This study considers all metrics equally important, with each weight assigned to the same value.

Once the figure of merit for solver performance evaluation is defined, a policy for handling cases of *parity scores* among the solvers must be established. One approach is to develop a multi-label, multi-class classifier, enabling the assignment of multiple labels to dataset items. Alternatively, a priority map among the solvers can be defined based on their characteristics. For simplicity in the current implementation, the latter approach is adopted. The *preference criterium* is based on the *number of qubits* involved in the solver and a preference for quantum solutions over the classical one. Specifically, the priority order is as follows:

1) QAOA, involving a number of qubits equal to the number of QUBO variables;
2) VQE, also involving a number of qubits equal to the number of QUBO variables;
3) GAS, involving a number of qubits equal to the number of QUBO variables plus the number of bits for the function operative range representation;
4) QA, involving multiple qubits for each binary variable due to minor embedding;
5) SA, which is the unique classical alternative considered.

In establishing the priority order, we have considered QAOA before VQE, even if they required the same number of qubits, based on empirical observations. Indeed, we have noticed that, on average, QAOA reaches convergence faster than VQE.

2) Identification of the Problems Features: Afterwards, it is necessary to define QUBO information with minimal computational effort for their inclusion as *features* in the prediction model. Considering the characteristics of the QUBO solver, we proposed to employ the following features:

- number of variables in the QUBO problem;
- number of non-zeros first-order QUBO terms (a_i);
- number of non-zeros second-order QUBO coefficients (b_{ij});
- average of non-zeros first-order QUBO terms (a_i);
- variance of non-zeros first-order coefficients (a_i);
- average of non-zeros second-order QUBO coefficients (b_{ij});
- variance of non-zeros second-order QUBO terms (b_{ij});
- average of all coefficients (a_i and b_{ij});
- variance of all coefficients (a_i and b_{ij}).

One of the main challenges in this step lies in identifying problem characteristics that may affect solver performance and estimating them with limited computational resources. Specifically, we focused on the count of non-zero coefficients of the second order, as they influence the number of two-qubit relationships that need to be expressed on quantum computers. The total number of variables, as well as the counts of first and second-order elements, collectively determine the QUBO density, which affects the shape of the associated cost function. Moreover, the average and variance of the problem coefficients impact the fluctuations' width in the QUBO function, thus also influencing the effectiveness of various exploration strategies, as suggested in [10].

All this information can be easily extrapolated from the final QUBO formulation with computational complexity growing at most quadratically with the number of problem variables.

3) Creation of the Dataset: After the definition of evaluation metrics, priority map and features, it is necessary to generate a proper dataset for training and evaluating the classification models. More than *500 different QUBO problems* have been considered. In particular, we consider:

- the well-known benchmark of conventional optimization (those without non-linear elements) reported in [50] and written as QUBO;
- problems generated of different sizes, densities and range of coefficients from the QUBO formulations reported in [18];

- problems generated of different sizes, densities and range of coefficients from the QUBO formulations reported in [51];
- portfolio optimization problem, written as QUBO;
- linear regression problem on the Iris dataset written as QUBO.

These benchmarks have been carefully selected to encompass a wide range of optimization problems that users may encounter, providing a comprehensive overview of potential scenarios of interest.

4) Preprocessing Techniques: The resulting dataset exhibits class *imbalance*, meaning that the proportion of samples belonging to each class changes. In particular, QAOA emerges as the best solver for little less than half of the entire dataset, followed by QA and VQE, each representing approximately 20%, while GAS and SA are optimal for around 10% of the analyzed problems. Unfortunately, an unbalanced classification task is more complex than a balanced one [52].

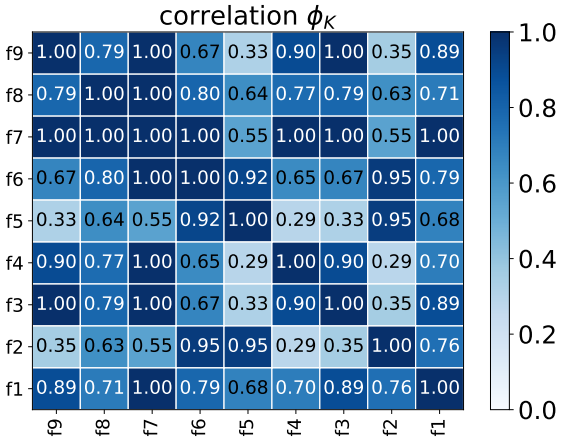


Fig. 8: ϕ_K correlation matrix defined in [53]. It expresses also non-linear relations among the features.

Some *pre-processing* techniques can be applied to improve model stability and *reduce the dataset's redundancy and dimensionality*, mitigating the task complexity. First of all, features have been scaled to a common range to avoid model domination by the larger scale features and to remove model bias. Moreover, in this work, we evaluate the potential of the *Principal Component Analysis* (PCA, [54]) and *Linear Discriminant Analysis* (LDA, [55]) techniques. PCA, an unsupervised learning method, identifies the principal components, representing orthogonal directions of maximum variance, to capture the dataset's greatest variability. On the other hand, LDA, a supervised learning technique, computes the optimal data projection onto a lower-dimensional subspace, maximizing class separability. Both techniques aim to extract features while retaining crucial information in the data.

In this study, we explore datasets that have been dimensionally reduced using PCA with 2, 3, 4 (retaining 99% of data variability), and 9 components, alongside LDA with 2, 3, and 4 elements. Examination of the ϕ_K correlation matrix presented in Figure 8 suggests that PCA and LDA could have a positive impact on model performance since the dataset presents redundancy and relations among features.

5) Supervised Learning Models: For the predictor development, a five-fold cross-validation technique has been exploited, considering the following supervised machine-learning classifiers of the scikit-learn library [56], properly tuning their degrees of freedom:

- *Ada Boost* [57];
- *Decision Tree* [58];
- *Gradient Boosting* [59];
- *k-nearest neighbors* (KNN, [60]);
- *Logistic Regression* [61];
- *Naive Bayes* [61];
- *Neural Network* [62];
- Random Forest [63];
- *Support Vector Machine* (SVM, [64]);
- *eXtreme Gradient Boosting* (XGBoost [65]).

To address the dataset's imbalance, the creation of the folds has exploited a *stratification technique* for maintaining the proportion among elements belonging to each class.

The k cross-validation approach is crucial for optimizing model performance and ensuring robustness and reliability in estimating the quality of solver prediction with unseen problems. Additionally, it enhances efficiency in comparing the generalization capabilities of different models.

B. Solvers Parameter Setting

In this section, simple strategies for effectively configuring and scaling the parameters of quantum solvers are proposed. These options are those considered for the predictive model dataset generation and, consequently, for each new problem for configuring the selected solver. Specifically, for the dataset creation, the *Advantage_system4.1* D-Wave device has been chosen for QA, while the Qiskit implementation of QAOA, VQE, and GAS solvers, launched on the QASM simulator, has been employed for their respective configurations. Finally, the D-Wave implementation of the SA solver has been employed.

1) Quantum Annealer: Defining the *annealing time* scaling with the QUBO problem size is crucial for QA. This relation can be deduced from state-of-the-art plots representing the *time-to-solution* (TTS) metric relative to the problem size. In [66], it is proved that the annealing time necessary is expected to grow as

$$T_{\text{TTS}} = 10^{b\sqrt{N}}, \tag{3}$$

where N is the number of variables, while b is a coefficient. From plots in the literature [66] and our experience, b equals 0.7 is a reasonable value.

2) Quantum Approximate Optimization Algorithm: The parameter most critical to scale with the problem size in QAOA is the number of *repetitions*, as discussed in Section II-B. Also in this case, the scaling function for this parameter has been derived from literature plots, particularly those found in [48], [49]. Our analysis reveals that the number of repetitions required for achieving satisfactory results increases proportionally to $\sqrt{N}$, where N represents the number of problem variables. The relation adopted for dataset creation is given by:

$$\text{reps} = \left\lceil 2\sqrt{N} \right\rceil .$$

Moreover, we have considered Hadamard state initialization and $-X$ mixed state, as in Figure 2. For classical optimization, the Cobyla optimizer, providing on average good performance, has been selected without imposing restrictions on the number of classical iterations, allowing the algorithm to automatically stop when the convergence is achieved.

3) Variational Quantum Eigensolver: Unlike the other solver, VQE has no critical parameters to be set scaling with the problem size. For dataset creation, an ansatz composed of a Ry gate layer and an entangling layer, like that shown in Figure 3, has been chosen. Similarly to QAOA, for classical optimization, Cobyla, providing on average good performance, has been exploited without a restriction on the number of classical iterations, allowing the algorithm to terminate upon convergence.

4) Grover Adaptive Search: The GAS parameter setting is very complex, involving the determination of the algorithm's stop *threshold*, which scales with the problem size, and the selection of the *number of qubits for representing cost function values*, dependent on the specific problem.

Unfortunately, analyses for threshold scaling are lacking in the literature. Therefore, we have empirically discovered that a linear growth with the problem size provides reasonable results quality, and it is the same scaling adopted in [31] with the Qiskit framework. In particular, for dataset creation, the following scaling function has been considered:

$$\text{th} = 2N\,, \tag{4}$$

where N is the number of problem variables.

For what concerns cost function values encoding, we propose to leverage the estimation of the *functions bound* based on the exploitation of *posiform* and *negaform* proposed in [67], [68] after the application of a preprocessing step. Considering that the image of the cost function is represented in fixed point, for saving qubits, we choose first to round the problem coefficients to a user-selected precision, then *normalize* all QUBO matrix elements such that the smallest one (in terms of absolute value) is equal to 1. In this way, the waste in terms of the number of value qubits should be minimized.

5) Simulated Annealing: Similarly to QA, for SA, the crucial parameter to scale with the problem size is the *number of iterations* (or sweeps), which is strongly related to annealing time. Also in this case, TTS can be exploited, and the same type of scaling of QA (Equation 3) is expected from analysis in [66]. Empirically, we found that b equals 0.5 is a reasonable value.

V. Results

The supervised learning models, presented in Section 5, have been trained and evaluated with and without applying the features reduction technique discussed in Section 4 exploiting a five-fold cross-validation procedure on a dataset including more than 500 different QUBO problems. The classifier and feature reduction techniques quality are compared in Table I in terms of:

- *Accuracy*, which is the percentage of correct predictions found with the k cross-validation approach.

TABLE I: Performance comparison across various classifiers and feature reduction methods, highlighting accuracy (Acc), the percentage of top two predictions, and the average error in the success probability (p_s err). The best results are highlighted in **bold** and green.

Models		No preproc	PCA				LDA		
			2	3	4	9	2	3	4
Ada Boost	Accuracy [%]	64.48	55.79	59.78	62.68	62.85	64.30	64.30	65.21
	Top two [%]	86.23	81.70	82.07	85.14	85.33	84.96	84.78	84.96
	p_s err [%]	4.26	6.90	6.87	7.22	7.42	7.31	7.57	6.91
Decision Tree	Accuracy [%]	68.65	64.85	65.75	63.57	67.02	65.75	65.75	65.93
	Top two [%]	87.50	84.60	85.87	84.78	85.14	85.33	85.33	85.69
	p_s err [%]	3.22	4.79	4.45	4.45	4.91	4.80	4.31	4.42
Gradient Boosting	Accuracy [%]	72.63	67.02	67.75	69.20	69.37	66.66	66.29	66.65
	Top two [%]	89.86	86.41	87.50	86.96	86.59	85.87	86.05	86.78
	p_s err [%]	2.40	4.37	4.67	4.15	4.37	4.80	4.64	3.89
KNN	Accuracy [%]	57.79	56.52	56.88	57.42	57.42	59.24	59.24	59.96
	Top two [%]	81.52	78.08	78.44	79.89	79.89	82.79	82.25	82.79
	p_s err [%]	7.37	7.07	7.18	7.33	7.33	6.86	6.97	6.92
Logistic Regression	Accuracy [%]	71.01	32.06	42.57	50.18	50.17	53.62	55.06	55.43
	Top two [%]	88.59	55.25	70.29	78.26	77.72	78.26	80.80	81.70
	p_s err [%]	3.70	13.32	9.62	7.64	8.17	9.96	6.60	6.31
Naive Bayes	Accuracy [%]	53.09	56.52	59.41	61.58	60.49	60.68	60.32	60.68
	Top two [%]	77.36	78.08	81.52	83.88	84.06	82.43	82.43	83.51
	p_s err [%]	7.49	7.07	7.02	6.70	7.37	8.66	8.05	7.89
Neural Network	Accuracy [%]	57.78	63.76	63.94	66.12	64.49	63.04	64.48	63.58
	Top two [%]	81.16	85.51	87.14	85.87	86.23	87.50	87.32	86.59
	p_s err [%]	6.63	5.69	6.52	6.73	7.13	6.50	6.90	6.81
Random Forest	Accuracy [%]	**73.18**	66.30	68.46	69.92	68.83	69.56	68.29	68.65
	Top two [%]	**91.12**	86.05	87.68	88.95	87.50	88.59	87.32	87.32
	p_s err [%]	**2.16**	4.06	3.80	3.17	3.87	3.70	3.80	3.94
SVM	Accuracy [%]	59.06	56.52	61.04	62.68	62.68	63.04	64.48	63.94
	Top two [%]	83.70	78.08	84.24	85.69	85.69	86.96	87.14	86.59
	p_s err [%]	7.17	7.07	6.52	6.62	6.62	6.55	6.46	6.43
XGBoost	Accuracy [%]	69.56	66.30	67.02	68.46	66.84	66.12	67.19	67.02
	Top two [%]	87.50	85.69	86.23	87.50	84.96	87.14	87.50	87.50
	p_s err [%]	3.01	4.12	4.57	3.66	4.61	4.47	4.24	4.05

- *Top 2*, which is the relative frequency of predicting one of the top two solvers.
- *Average p_s error*, which is the average distance between the probability of achieving the optimal solution of the best solver and of the predicted solver.

We choose to present the average error in success probability rather than those in the overall score because it offers a more explicit interpretation with respect to an abstract score. Furthermore, its range of possible values is problem-independent, unlike the score—dependent on function bounds—, thus simplifying the process of averaging the results.

The effectiveness of the features reduction technique depends on the type of model. For example, considering the model SVM, Naive Bayes, or Neural Network, they are particularly useful, while in Random Forest, they do not provide benefit.

Observing the table, it is clear that Random Forest outperforms other supervised learning models in predicting the expected optimal solver. It achieves the best performance with an accuracy of 73.18% and about 90% rate of predictions providing a solver in the top two. Additionally, the average loss

in terms of the probability of obtaining the optimal solution is almost negligible, at just 2.16%. These results have been obtained with the following hyperparameter setting:

- number of Decision Trees equal to 100;
- maximal depth equal to 50;
- minimal samples per leaf equal to 1;
- minimal samples per split equal to 2.

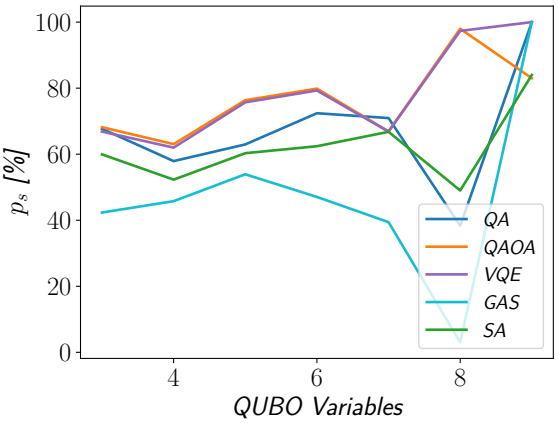


Fig. 9: The average probability of achieving the expected optimal value (p_s) across the dataset's problems is plotted against the number of QUBO variables.

Finally, Figure 9 shows the average probability of achieving the optimal solution of the problems considered in the dataset creation across their sizes. These problems, as previously mentioned, exhibit considerable variations in density, dimension, and variance of the involved coefficients, etc. It is possible to notice the consistency in solution quality across different problem sizes, proving the reasonableness of the proposed parmeters scaling approach.

VI. Discussion

The conducted exploration demonstrates the feasibility of approaching solver selection as a classification task. However, a significant challenge lies in the necessity of a large dataset, which is computationally and economically expensive to obtain, to ensure a reliable prediction. While the dataset dimensions in this manuscript suffice to establish proof of concept, providing encouraging results, more training data are needed to achieve completely satisfactory prediction results. Expanding the dataset poses challenges related to the selection of diverse problems to provide a comprehensive overview of potential scenarios. Furthermore, the maximum size of the problems that can be addressed is constrained by the limits of current quantum computer simulators, especially for GAS execution, requiring the use of complex High-Performance Computing systems to be able to consider the most interesting problems.

Moreover, while the chosen solvers' settings for dataset creation have demonstrated their reasonableness, relying exclusively on empirical deductions is not the optimal approach. Consequently, there is potential for significant enhancements in the implementation of predictors.

VII. Conclusions

This work proposes to address the *solver selection* challenge with a *supervised machine learning approach*, treating it as a classification task. Moreover, the article suggests strategies for *adjusting solver parameters* based on problem size and characteristics. To this end, we first reviewed quantum optimization, focusing on solvers and flow required for solving an optimization problem with quantum computers. Then, we delve into the motivation driving this exploration and the objectives of the research. Afterwards, the methodology followed for developing the solver predictor is outlined step by step, explaining and motivating design choices, and the solvers' setting, based on state-of-the-art experience and considered for dataset creation, has been introduced. The effectiveness of the supervised learning approach is validated through experimentation with a *dataset comprising 500 diverse QUBO problems*.

The pre-trained classifier is integrated into the MQT Quantum Auto Optimizer (MQT QAO) framework, publicly available on GitHub (https://github.com/cda-tum/mqt-qao) as part of the Munich Quantum Toolkit (MQT).

Even though the obtained results are promising, opportunities for enhancement and expansion remain. First, the proposed parameters' setting can be improved by involving machine learning models. Moreover, the solver selection predictor can be improved both enlarging the considered training dataset and considering more quantum and classical solvers and their variances. In addition, a multi-label approach for managing the parity cases, instead of the priority mechanism, can be considered, and reinforcement learning can be evaluated as an alternative to supervised learning.

In conclusion, this research proves the potential of machine learning in quantum solver selection, offering valuable tools for non-experts in quantum computing. We hope that this exploration will be the starting point towards the development of automated instruments for managing quantum solvers, thereby enabling the creation of quantum solutions for real-world problems by a broader spectrum of users.

Acknowledgements

N.Q. and R.W. acknowledge funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 101001318), the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the High-tech Agenda Bayern Plus, and the BMWK on the basis of a decision by the German Bundestag through project QuaST, as well as the BMK, BMDW, the State of Upper Austria in the frame of the COMET program, and the QuantumReady project within Quantum Austria (managed by the FFG).
D.V. would like to thank HPC@POLITO — a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://hpc.polito.it) — for providing computational resources.

References

[1] J. Lang, S. Zielinski, and S. Feld, "Strategic portfolio optimization using simulated, digital, and quantum annealing," *Applied Sciences*, vol. 12, no. 23, p. 12288, 2022. https://doi.org/10.3390/app122312288.

[2] T. Ohyama, Y. Kawamoto, and N. Kato, "Resource allocation optimization by quantum computing for shared use of standalone irs," *IEEE Transactions on Emerging Topics in Computing*, 2023. https://doi.org/10.1109/TETC.2023.3292355.

[3] J. Zhang, G. L. Bianco, and J. C. Beck, "Solving job-shop scheduling problems with qubo-based specialized hardware," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 32, pp. 404–412, 2022. https://doi.org/10.1609/icaps.v32i1.19826.

[4] autoqubo GitHub repository, "autoqubo github repository." Accessed: December 15, 2023.

[5] A. Moraglio, S. Georgescu, and P. Sadowski, "AutoQubo: data-driven automatic QUBO generation," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 2232–2239, 2022. https://doi.org/10.1145/3520304.3533965.

[6] J. Pauckert, M. Ayodele, M. D. García, S. Georgescu, and M. Parizy, "Autoqubo v2: Towards efficient and effective qubo formulations for ising machines," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pp. 227–230, 2023. https://doi.org/10.1145/3583133.3590662.

[7] P. M. Xavier, P. Ripper, T. Andrade, J. D. Garcia, N. Maculan, and D. E. B. Neira, "Qubo.jl: A julia ecosystem for quadratic unconstrained binary optimization," *arXiv preprint arXiv:2307.02577*, 2023. https://doi.org/10.48550/arXiv.2307.02577.

[8] D. Volpe, N. Quetschlich, M. Graziano, G. Turvani, and R. Wille, "Towards an automatic framework for solving optimization problems with quantum computers," *2024 IEEE International Conference on Quantum Software (QSW)*, 2024.

[9] N. Quetschlich, L. Burgholzer, and R. Wille, "Towards an automated framework for realizing quantum computing solutions," in *2023 IEEE 53rd International Symposium on Multiple-Valued Logic (ISMVL)*, pp. 134–140, IEEE, 2023.

[10] N. Chancellor, "Modernizing quantum annealing using local searches," *New Journal of Physics*, vol. 19, no. 2, p. 023024, 2017. https://doi.org/10.1088/1367-2630/aa59c4.

[11] R. Wille, L. Berent, T. Forster, J. Kunasaikaran, K. Mato, T. Peham, N. Quetschlich, D. Rovara, A. Sander, L. Schmid, D. Schoenberger, Y. Stade, and L. Burgholzer, "The MQT Handbook: A Summary of Design Automation Tools and Software for Quantum Computing," in *IEEE International Conference on Quantum Software (QSW)*.

[12] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," *Physical Review E*, vol. 58, no. 5, p. 5355, 1998. https://doi.org/10.1103/PhysRevE.58.5355.

[13] B. K. Chakrabarti, H. Leschke, P. Ray, T. Shirai, and S. Tanaka, "Quantum annealing and computation: challenges and perspectives," *Philosophical Transactions of the Royal Society A*, vol. 381, no. 2241, p. 20210419, 2023. https://doi.org/10.1098/rsta.2021.0419.

[14] A. Rajak, S. Suzuki, A. Dutta, and B. K. Chakrabarti, "Quantum annealing: an overview," *Philosophical Transactions of the Royal Society A*, vol. 381, no. 2241, p. 20210417, 2023. https://doi.org/10.1098/rsta.2021.0417.

[15] D. Venturelli, S. Mandrà, S. Knysh, B. O'Gorman, R. Biswas, and V. Smelyanskiy, "Quantum optimization of fully connected spin glasses," *Physical Review X*, vol. 5, no. 3, p. 031040, 2015. https://doi.org/10.1103/PhysRevX.5.031040.

[16] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011. https://doi.org/10.1038/nature10012.

[17] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge ; New York: Cambridge University Press, 10th anniversary ed ed., 2010.

[18] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using qubo models," *arXiv preprint arXiv:1811.11538*, 2018. https://doi.org/10.48550/arXiv.1811.11538.

[19] E. Combarro, S. Gonzalez-Castillo, and A. Di Meglio, "A practical guide to quantum machine learning and quantum optimization: Hands-on approach to modern quantum algorithms," 2023.

[20] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, May 1983. https://doi.org/10.1126/science.220.4598.671.

[21] M. S. Könz, W. Lechner, H. G. Katzgraber, and M. Troyer, "Embedding overhead scaling of optimization problems in quantum annealing," *PRX Quantum*, vol. 2, no. 4, p. 040322, 2021. https://doi.org/10.1103/PRXQuantum.2.040322.

[22] 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," *arXiv preprint arXiv:2306.09198*, 2023. https://doi.org/10.48550/arXiv.2306.09198.

[23] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014. https://doi.org/10.48550/arXiv.1411.4028.

[24] M. Suzuki, "Generalized trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems," *Communications in Mathematical Physics*, vol. 51, no. 2, pp. 183–190, 1976.

[25] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth, *et al.*, "The variational quantum eigensolver: a review of methods and best practices," *Physics Reports*, vol. 986, pp. 1–128, 2022. https://doi.org/10.1016/j.physrep.2022.08.003.

[26] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, p. 4213, 2014. https://doi.org/10.1038/ncomms5213.

[27] D. Bulger, W. P. Baritompa, and G. R. Wood, "Implementing pure adaptive search with grover's quantum algorithm," *Journal of optimization theory and applications*, vol. 116, pp. 517–529, 2003. https://doi.org/10.1023/A:1023061218864.

[28] A. Gilliam, S. Woerner, and C. Gonciulea, "Grover adaptive search for constrained polynomial binary optimization," *Quantum*, vol. 5, p. 428, 2021. https://doi.org/10.22331/q-2021-04-08-428.

[29] Y. Sano, K. Mitarai, N. Yamamoto, and N. Ishikawa, "Accelerating grover adaptive search: Qubit and gate count reduction strategies with higher-order formulations," *arXiv preprint arXiv:2308.01572*, 2023. https://doi.org/10.48550/arXiv.2308.01572.

[30] Y. Sano, M. Norimoto, and N. Ishikawa, "Qubit reduction and quantum speedup for wireless channel assignment problem," *IEEE Transactions on Quantum Engineering*, 2023. https://doi.org/10.1109/TQE.2023.3293452.

[31] L. Giuffrida, D. Volpe, G. A. Cirillo, M. Zamboni, and G. Turvani, "Engineering grover adaptive search: Exploring the degrees of freedom for efficient qubo solving," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 3, pp. 614–623, 2022. https://doi.org/10.1109/JETCAS.2022.3202566.

[32] W. P. Baritompa, D. W. Bulger, and G. R. Wood, "Grover's quantum algorithm applied to global optimization," *SIAM Journal on Optimization*, vol. 15, no. 4, pp. 1170–1184, 2005. https://doi.org/10.1137/040605072.

[33] M. Fingerhuth, T. Babej, and P. Wittek, "Open source software in quantum computing," *PloS one*, vol. 13, no. 12, p. e0208561, 2018. https://doi.org/10.1371/journal.pone.0208561.

[34] G. G. Guerreschi, J. Hogaboam, F. Baruffa, and N. P. Sawaya, "Intel quantum simulator: A cloud-ready high-performance simulator of quantum circuits," *Quantum Science and Technology*, vol. 5, no. 3, p. 034007, 2020. https://doi.org/10.1088/2058-9565/ab8505.

[35] A. Zulehner and R. Wille, "Advanced simulation of quantum computations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 5, pp. 848–859, 2018. https://doi.org/10.1109/TCAD.2018.2834427.

[36] S. Hillmich, I. L. Markov, and R. Wille, "Just like the real thing: Fast weak simulation of quantum computation," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2020. https://doi.org/10.1109/DAC18072.2020.9218555.

[37] T. Vincent, L. J. O'Riordan, M. Andrenkov, J. Brown, N. Killoran, H. Qi, and I. Dhand, "Jet: Fast quantum circuit simulations with parallel task-based tensor-network contraction," *Quantum*, vol. 6, p. 709, 2022. https://doi.org/10.22331/q-2022-05-09-709.

[38] B. Villalonga, S. Boixo, B. Nelson, C. Henze, E. Rieffel, R. Biswas, and S. Mandrà, "A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware," *npj Quantum Information*, vol. 5, no. 1, p. 86, 2019. https://doi.org/10.1038/s41534-019-0196-1.

[39] P. Documentation, "Pyqubo documentation - getting started." Accessed: December 15, 2023.

[40] M. Zaman, K. Tanahashi, and S. Tanaka, "Pyqubo: Python library for mapping combinatorial optimization problems to qubo form," *IEEE Transactions on Computers*, vol. 71, no. 4, pp. 838–850, 2021. https://doi.org/10.1109/TC.2021.3063618.

[41] qubovert Documentation, "qubovert documentation - getting started." Accessed: December 15, 2023.

[42] dimod Documentation, "dimod documentation - getting started." Accessed: December 15, 2023.

[43] qiskit-optimization Documentation, "qiskit-optimization documentation - getting started." Accessed: December 15, 2023.

[44] fixstarts Documentation, "fixstarts documentation - getting started." Accessed: December 15, 2023.

[45] openQAOA entropica labs Documentation, "openqaoa entropica labs documentation - getting started." Accessed: December 15, 2023.

[46] N. Quetschlich, L. Burgholzer, and R. Wille, "MQT Predictor: Automatic Device Selection with Device-Specific Circuit Compilation for Quantum Computing," *ACM Transactions on Quantum Computing (TQC)*, 2024.

[47] S. P. Adam, S.-A. N. Alexandropoulos, P. M. Pardalos, and M. N. Vrahatis, "No free lunch theorem: A review," *Approximation and optimization: Algorithms, complexity and applications*, pp. 57–82, 2019.

[48] X. Lee, Y. Saito, D. Cai, and N. Asai, "Parameters fixing strategy for quantum approximate optimization algorithm," in *2021 IEEE international conference on quantum computing and engineering (QCE)*, pp. 10–16, IEEE, 2021.

[49] M. Y. Niu, S. Lu, and I. L. Chuang, "Optimizing qaoa: Success probability and runtime dependence on circuit depth," *arXiv preprint arXiv:1905.12134*, 2019. https://doi.org/10.48550/arXiv.1905.12134.

[50] "Surjanovic, s., & bingham, d. (2013). virtual library of simulation experiments: Test functions and datasets. retrieved from https://www.sfu.ca/ ssurjano/optimization.html." Online, 2013.

[51] A. Lucas, "Ising formulations of many np problems," *Frontiers in physics*, vol. 2, p. 74887, 2014. https://doi.org/10.3389/fphy.2014.00005.

[52] J. Brownlee, *Imbalanced classification with Python: better metrics, balance skewed classes, cost-sensitive learning*. Machine Learning Mastery, 2020.

[53] M. Baak, R. Koopman, H. Snoek, and S. Klous, "A new correlation coefficient between categorical, ordinal and interval variables with pearson characteristics," *Computational Statistics & Data Analysis*, vol. 152, p. 107043, 2020. https://doi.org/10.1016/j.csda.2020.107043.

[54] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016. https://doi.org/10.1098/rsta.2015.0202.

[55] A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien, "Linear discriminant analysis: A detailed tutorial," *AI communications*, vol. 30, no. 2, pp. 169–190, 2017. https://doi.org/10.3233/AIC-170729.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[57] A. J. Ferreira and M. A. Figueiredo, "Boosting algorithms: A review of methods, theory, and applications," *Ensemble machine learning: Methods and applications*, pp. 35–85, 2012. https://doi.org/10.1007/978-1-4419-9326-7_2.

[58] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991. https://doi.org/10.1109/21.97458.

[59] Z. He, D. Lin, T. Lau, and M. Wu, "Gradient boosting machine: a survey," *arXiv preprint arXiv:1908.06951*, 2019. https://doi.org/10.48550/arXiv.1908.06951.

[60] N. Bhatia *et al.*, "Survey of nearest neighbor techniques," *arXiv preprint arXiv:1007.0085*, 2010. https://doi.org/10.48550/arXiv.1007.0085.

[61] K. Das and R. N. Behera, "A survey on machine learning: concept, algorithms and applications," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, no. 2, pp. 1301–1309, 2017.

[62] S. Smys, J. I. Z. Chen, and S. Shakya, "Survey on neural network architectures with deep learning," *Journal of Soft Computing Paradigm (JSCP)*, vol. 2, no. 03, pp. 186–194, 2020.

[63] M. Zakariah *et al.*, "Classification of large datasets using random forest algorithm in various applications: Survey," *International Journal of Engineering and Innovative Technology (IJJEIT)*, vol. 4, no. 3, 2014.

[64] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, pp. 189–215, 2020.

[65] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016. https://doi.org/10.1145/2939672.2939785.

[66] S. Mandra, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber, "Strengths and weaknesses of weak-strong cluster problems: A detailed overview of state-of-the-art classical heuristics versus quantum approaches," *Physical Review A*, vol. 94, no. 2, p. 022337, 2016. https://doi.org/10.1103/PhysRevA.94.022337.

[67] E. Boros and P. L. Hammer, "Pseudo-boolean optimization," *Discrete applied mathematics*, vol. 123, no. 1-3, pp. 155–225, 2002. https://doi.org/10.1016/S0166-218X(01)00341-9.

[68] E. Boros, P. L. Hammer, and G. Tavares, "Preprocessing of unconstrained quadratic binary optimization," tech. rep., Technical Report RRR 10-2006, RUTCOR, 2006.