Quantum Solution for Configuration Selection and Prioritization

Joshua Ammermann
joshua.ammermann@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Baden-Wuerttemberg, Germany

Fabian Jakob Brenneisen
fabian.brenneisen@student.kit.edu
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Baden-Wuerttemberg, Germany

Tim Bittner
tim.bittner@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Baden-Wuerttemberg, Germany

Ina Schaefer
ina.schaefer@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Baden-Wuerttemberg, Germany

ABSTRACT

The analyses of highly configurable systems, as applied in software or automotive domains, yield hard problems due to the exponentially increasing number of possible product configurations. Current research identified that such combinatorial optimization problems, e.g. configuration selection and prioritization, are ideal targets for expected exponential quantum speedups. However, empirical evidence about the applicability of quantum computing to these problems is still missing. In this paper, we investigate how the constraint satisfaction and optimization problems of configuration selection and prioritization can be addressed using quantum computing. We propose a method to transform the configuration selection and prioritization problems encoded in attributed feature models into a quantum mechanical formulation suitable for optimization problems. We provide a Python library to automatically perform this transformation and apply the Quantum Approximate Optimization Algorithm (QAOA), such that configuration selection and prioritization are solved with quantum computers. Our approach is evaluated regarding feasibility, solution quality, and scalability. We show that QAOA obtains good results regarding configuration selection, but for configuration prioritization, the approach needs further improvement.

CCS CONCEPTS

• **Software and its engineering** → **Software configuration management and version control systems**; • **Hardware** → **Quantum computation**.

KEYWORDS

Configuration Selection, Configuration Prioritization, QAOA

ACM Reference Format:
Joshua Ammermann, Fabian Jakob Brenneisen, Tim Bittner, and Ina Schaefer. 2024. Quantum Solution for Configuration Selection and Prioritization. In *2024 ACM/IEEE International Workshop on Quantum Software Engineering (Q-SE 2024), April 16, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3643667.3648221

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

ACM ISBN 979-8-4007-0570-0/24/04.
https://doi.org/10.1145/3643667.3648221

1 INTRODUCTION

Quantum algorithms with proven theoretical speedups exist, such as a quadratic speedup with Grover's algorithm [10] and an exponential speedup with Shor's algorithm [18]. However, current quantum computers in the Noisy Intermediate-Scale Quantum (NISQ) era face challenges regarding their scalability. Nevertheless, there exist hybrid optimization algorithms that split the workload between classical and quantum machines, which can be applied even in the NISQ era. Current research on the application of quantum algorithms aims to achieve quantum readiness, so that once quantum hardware matures, its benefits can directly be leveraged.

The analyses of highly configurable systems, as applied in software or automotive domains, may be improved using quantum computing [6]. Problems in these domains are hard to solve due to the exponentially increasing number of possible product configurations (configuration space explosion). Eichhorn et al. [6] gives an overview of analyses for configurable systems that may be improved using quantum computing to identify potentials and challenges. They identified that Boolean satisfiability (SAT) and ILP-based feature model analysis could potentially be sped up using quantum algorithms like Grover and the Quantum Approximate Optimization Algorithm (QAOA) [7]. However, empirical evidence about the applicability of quantum computing to these problems is still missing. Two concrete problems of configuration selection and prioritization are constraint satisfaction (SAT) and optimization problems. These problems are promising candidates for hybrid quantum-classical optimization as they are classically addressed using approximation algorithms.

In this paper, we investigate if and how the constraint satisfaction and optimization problems of configuration selection and prioritization can be addressed using quantum computing. We contribute a method to transform the problems encoded in attributed feature models into a quantum mechanical formulation suitable for optimization problems. We provide a Python library to automatically perform this transformation and execute the Quantum Approximate Optimization Algorithm (QAOA), such that configuration selection and prioritization are solved with quantum computers. We evaluate whether QAOA is suitable for these problems (feasibility), what quality of results are obtained for small problem instances (solution quality), and how the quantum hardware requirements scale with problem instances (scalability). We show that QAOA obtains good results regarding configuration selection, but for configuration prioritization, the approach needs further improvement.

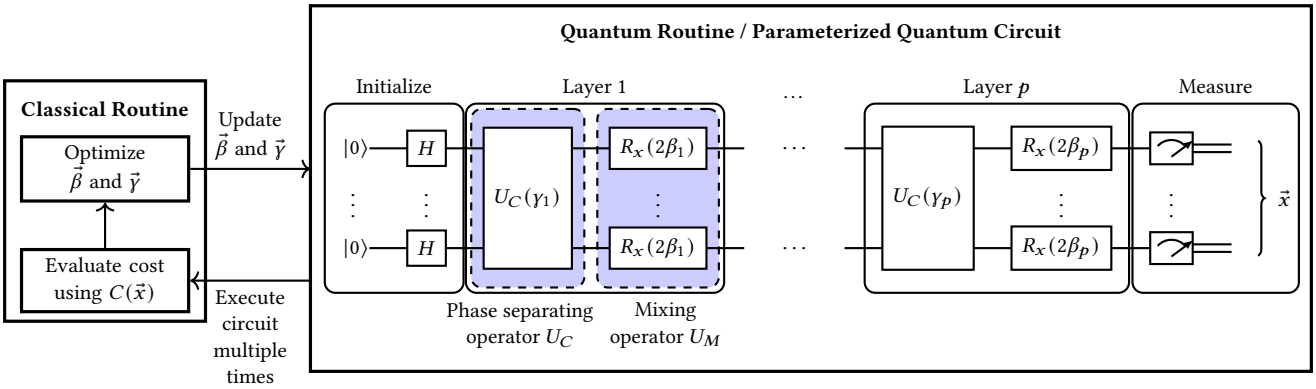


Figure 1: Circuit representation of the QAOA classical and quantum routine used in this work. The quantum circuit is parameterized by two parameter vectors $\vec{\beta} = \beta_1, ..., \beta_p$ and $\vec{\gamma} = \gamma_1, ..., \gamma_p$, where p denotes the number of layers of QAOA. The operator U_C is constructed for each problem instance according to the cost function of the optimization problem $C(\vec{x})$.

2 BACKGROUND

We introduce relevant quantum computing background and refer the interested reader to the book by Nielsen and Chuang [15].

2.1 Quantum Computing

Quantum Computing utilizes quantum mechanical effects for computation. In contrast to classical computers, whose basic unit of computation is a bit in one of the two states 0 and 1, the basic unit of quantum computation is a *qubit*. A qubit can be in a superposition of two orthonormal basis states, e.g., the computational basis states $|0\rangle$ and $|1\rangle$: $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$, where $|\psi\rangle$ denotes the overall quantum state of the qubit. The factors $a, b \in \mathbb{C}$ are the probability amplitudes and the equality $|a|^2 + |b|^2 = 1$ holds. Measuring a qubit is a destructive operation that collapses the qubit's quantum state into a classical bit with the probability of $|a|^2$ to 0 and $|b|^2$ to 1. Reversible unitary operations U can be applied to qubits to change their state, these are also referred to as *gates*. Commonly used single qubit gates relevant for this work are the Hadamard gate H, and the parameterized rotation gates $R_x(\theta)$ and $R_z(\theta)$. The Hadamard gate H is referred to as the superposition gate, as it puts a qubit in state $|0\rangle$ into the *equal superposition* state $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, and a qubit in state $|1\rangle$ to state $|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$ respectively. The rotation gates $R_x(\theta)$ and $R_z(\theta)$ perform a rotation of θ around the x and z-axis on the Bloch sphere respectively. Quantum gates can also be applied to multiple qubits and common multi-qubit gates are controlled gates. Controlled gates are often used to introduce *entanglement* between qubits, as they apply an operation on a target qubit iff one or multiple control qubits are in $|1\rangle$. E.g., multi-controlled R_z gates may apply R_z on the target qubit dependent on control qubits. The Ising spin model originally describes ferromagnetism using a polynomial function of N spins s_i that can be in the states +1 or −1. The Ising Hamiltonian computing the energy of a spin configuration $s_1, ..., s_N$ is given by Lucas [14]:

$$H(s_1, ..., s_N) = -\sum_{i<j} J_{ij} s_i s_j - \sum_{i=1}^{N} h_i s_i,$$

where the coefficients J_{ij} and field h_i are real numbers. A quantum version of this Hamiltonian can be obtained by using Pauli-Z matrices σ_i^z for the spins s_i [14]. These Ising Hamiltonians will be used as Cost Hamiltonians in QAOA.

2.2 Quantum Approximate Optimization Algorithm (QAOA)

The Quantum Approximate Optimization Algorithm (QAOA) [7] is a prominent NISQ algorithm that splits the workload between a classical and a quantum computer, which results in shorter, less error-prone quantum circuits. The quantum computer computes a task encoded into a parameterized quantum circuit, while on the classical machine, the results are evaluated to adjust the parameters of the circuit using a classical optimizer. QAOA is a hybrid approximation algorithm tailored to solve combinatorial optimization problems. Since its introduction, many variants of QAOA have emerged, on which a recent review by Blekos et al. [2] gives an overview. Fig. 1 depicts the QAOA quantum and classical routine used in this work. The overall goal is to find an optimal bitstring $\vec{x}$ according to the cost function $C(\vec{x})$, which is given by the specific problem.

In each iteration of QAOAalgorithm, the expectation value of observables $\vec{x}$ is determined by executing the quantum circuit multiple times and evaluating the cost of returned solutions using $C(\vec{x})$. Then a classical optimizer tries to select better parameters of the circuit for the next iteration. This is repeated until convergence or until another stop criterion is met.

The QAOA quantum circuit is parameterized by two parameter vectors $\vec{\beta} = \beta_1, ..., \beta_p$ and $\vec{\gamma} = \gamma_1, ..., \gamma_p$ containing angles β and γ for each layer p of the circuit. Each circuit is initialized in a uniform superposition state using H gates on each qubit. Then p layers of phase separating and mixing operators follow. The phase separating operators U_C encode the cost function $C(\vec{x})$ and are parameterized by $\vec{\gamma}$. On the circuit level, we use multi-controlled R_z gates for this problem-instance-specific encoding in U_C. Mixing operators U_M change the amplitudes of the solutions and are parameterized by $\vec{\beta}$. For this work, we use the original implementation of mixing

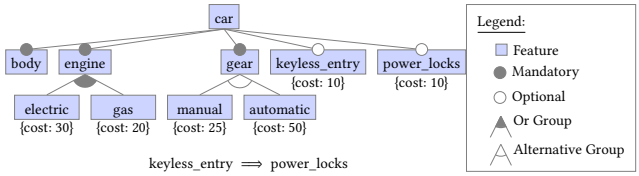


Figure 2: Example feature model representing configuration options of a car in the feature diagram notation with attributed feature costs.

operators using R_x gates on each qubit. Using R_z and R_x allows for arbitrary rotations around the Bloch sphere, while the controlled gates introduce entanglement. Repeated measurement returns the expectation value for the circuit with current parameters. By evaluating the cost function $C(\vec{x})$, the classical optimizer can make a selection for the parameters of the next iteration.

3 CONFIGURATION PROBLEMS

Analyzing highly configurable systems yields interesting problems due to the sheer amount of possible configurations. Each set of selected (and not selected) features represents a unique configuration of the system. Configurations are often represented in a graphical notation, a so-called feature model [12].

For this work, we use car configurations as our running example, which is depicted as a feature model in Fig. 2. All children of the abstract feature *car* are connected by an and-group. So all mandatory children *body*, *engine* and *gear* have to be selected in every valid configuration, but the selection of the features *keyless_entry* and *power_locks* is optional. The or-group denotes that at least one of the child features has to be selected (logical OR), e.g., the *engine* can either be selected as *electric*, *gas* or hybrid (*electric* and *gas*). If the children in a group are exclusive (logical XOR), e.g., a *gear* can either be *manual* or *automatic*, then the alternative group is used. Finally, cross-tree constraints in the form of a propositional formula can be added below the diagram, e.g., selecting *keyless_entry* implies that also *power_locks* have to be selected.

The feature model encodes constraints describing which configurations are valid. These constraints can be translated into a Boolean formula in Conjunctive Normal Form (CNF). For example, given the Boolean formula $(x_1 \vee x_2) \wedge x_3$ in CNF with the features x_1, x_2, x_3, then the bitstrings "101", "011" and "111" encode valid assignments or valid configurations respectively, in which a 1 denotes a selected and 0 a deselected feature. To obtain a minimal Boolean formula in CNF, we can omit features that always have to be selected (1) for a configuration to be valid: the root feature (e.g., *car*) and all mandatory child features that can be reached from the root (e.g., *body*, *engine* and *gear*), which can be identified by traversing the feature model. Such a minimal Boolean formula in CNF for the example is the following:

$$(electric \vee gas) \wedge (manual \vee automatic)$$
$$\wedge\ (\neg manual \vee \neg automatic) \wedge (\neg keyless\_entry \vee power\_locks)$$

From the 2^6 general possible product configurations over the 6 features, only 18 configurations are valid. Additionally, a variability model with n features $x_1, ..., x_n$ can assign each feature with attributes $a_1, ..., a_n$ (e.g., annotate each feature with a corresponding cost $c_1, ..., c_n$). For our example, the cost associated with each feature is depicted in Fig. 2. Such variability models can be represented by so-called attributed feature models [19].

In this work, we consider two Constraint Satisfaction and Optimization Problems (CSOPs) regarding the choice of configurations of a highly configurable system; configuration selection and configuration prioritization.

Problem 1: Configuration selection

Given FM attributed feature model is a set of $n \in \mathbb{N}$ features $FM = \{x_1, ..., x_n\}$ and each feature x_i contains an attribute a_i with $i \in [1, n]$
fc $\subseteq FM$ a configuration is a set of selected features
$C(fc)$: $fc \mapsto r \in \mathbb{R}$ cost function calculating the cost r of a configuration fc over the attributes a_j of features $\{x_j | x_j \in fc\}, j \leq n$

Goal Find a valid configuration according to FM that is optimal regarding $C(fc)$ with $fc \in FC$, where FC is the set all possible configurations.

E.g., using $\min C(x) = \sum_x c_i x_i$ to find the configuration with minimal costs. For our example, this would be the configuration "011000" with only the features *manual* and *gas* selected for an overall cost of 20 + 25 = 45.

Problem 2: Configuration prioritization

Given $FM, fc, C(fc)$ from Problem 1
$m \in \mathbb{N}$ number of searched configurations

Goal Find the optimal sequence of m valid configurations $[fc_0, ..., fc_m]$ according to FM, in which configurations are sorted by optimality in accordance with $C(fc)$: $C(fc_i) \leq C(fc_j), i < j$, and $i, j \in [0, m]$.

E.g., for finding the $m = 3$ lowest cost configurations, the desired solution for the running example would be the following sequence: ("011000", "011001", "101000"). Configurations in the sequence are ordered ascending by cost, so the first entry is again the minimal configuration configuration "011000". The second best configuration "011001" has an additional feature *power_locks* selected and a cost of 55 Also with a cost of 55 the third best configuration "101000" has the features *manual* and *electric* selected.

Both these problems are difficult to solve classically due to the number of possible product configurations, which grows exponentially with the number of features. A survey on classical solutions for product configuration [16] showed that evolutionary algorithms are most promising, even though the satisfaction of constraints cannot be guaranteed. As the best classical algorithmic solutions rely on approximation, this highlights the need to investigate approaches to address these problems with quantum computing to achieve quantum readiness.

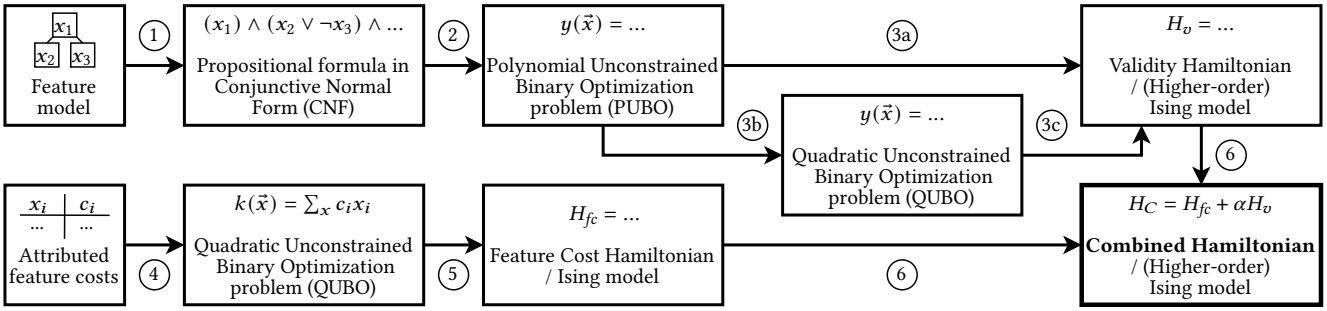


Figure 3: Transformation steps to translate an attributed feature model into a quantum mechanical formulation for QAOA.

4 QUANTUM SOLUTION FOR SELECTION AND PRIORITIZATION

To solve configuration selection and prioritization using quantum computing, we propose the following method. The method transforms a problem instance (an attributed feature model) into a quantum mechanical formulation for optimization problems - the Ising spin model. This problem formulation can then be used in QAOA as the phase separating operator U_C to run on a quantum computer. The proposed transformation process is divided into several transformation steps depicted in Fig 3. In general, we split our CSOP into the constraint satisfaction/validity part encoded in the feature model, and the optimization part encoded by the cost function. As an intermediate representation, we use unconstrained binary optimization problems, which are equivalent to Ising Hamiltonians [14]. We will now explain our method step-by-step.

4.1 Configuration Validity

The feature model encoding the validity of configurations is translated into a Boolean formula in CNF (1). Because this CNF representation can contain non-quadratic terms, it is translated into the form of a Polynomial Unconstrained Binary Optimization problem (PUBO) (2) - also called Higher-Order Unconstrained Binary Optimization problem (HUBO / HOBO) in the literature. This transformation can be realized directly with a generalization of the quadratic penalties described by Glover et al. [9] for the 2-SAT problem. One can transform an arbitrary clause formula in CNF $(x_i \vee \cdots \vee x_j \vee \neg x_k \vee \cdots \vee \neg x_l)$ into the clause penalty term $(1-x_i)\dots(1-x_j)(x_k)\dots(x_l)$. This term evaluates to 1 iff the clause is unsatisfied, which is the case if all negated features are selected and non-negated features are not selected, otherwise, it evaluates to 0. Computing these clause penalties for each clause and adding them, results in a penalty function $y(\vec{x})$ that calculates the overall penalties for the whole Boolean formula given a bitstring $\vec{x}$. Applied to the running example, this yields:

$$\begin{aligned} y(\vec{x}) &= (1-x_1)(1-x_2) + (x_3)(x_4) + (1-x_3)(1-x_4) + (x_5)(1-x_6) \\ &= 2 - x_1 - x_2 - x_3 - x_4 + x_5 + x_1x_2 + 2x_3x_4 - x_5x_6 \end{aligned}$$

The function $y(\vec{x})$ calculates the number of unsatisfied clauses. In other words, $y = 0$ indicates that all clauses are satisfied, which is desired for this problem, e.g. for $\vec{x}$ ="011000". Our example is a special case because it contains only quadratic terms, so it is also a Quadratic Unconstrained Binary Optimization problem (QUBO).

A PUBO can be transformed into a Hamiltonian in the Higher-order Ising spin model, and a QUBO into a Hamiltonian in the Ising spin model respectively. For this, the following rules have to be applied [21]:

- Rewrite variables x_i to $(1-z_i)/2$, where the new variables $z_i \in -1, 1$ instead of $x_i \in 0, 1$.
- Replace occurrences of z_i with Pauli operator σ_i^z.

The PUBO encoding the feature model constraints can be transformed into a Validity Hamiltonian H_v in the Higher-order Ising spin model (3a). For the running example, this results in:

$$\begin{aligned} y(\vec{z}) &= 2 - \frac{1-z_1}{2} - \frac{1-z_2}{2} - \frac{1-z_3}{2} - \frac{1-z_4}{2} + \frac{1-z_5}{2} \\ &\quad + \frac{1-z_1}{2} * \frac{1-z_2}{2} + 2 * \frac{1-z_3}{2} * \frac{1-z_4}{2} - \frac{1-z_5}{2} * \frac{1-z_6}{2} \\ &= 1 + z_1\frac{1}{4} + z_2\frac{1}{4} - z_5\frac{1}{4} + z_6\frac{1}{4} + z_1z_2\frac{1}{4} + z_3z_4\frac{1}{2} - z_5z_6\frac{1}{4} \quad \text{and} \end{aligned}$$

$$H_v = 1 * I + \sigma_1^z\frac{1}{4} + \sigma_2^z\frac{1}{4} - \sigma_5^z\frac{1}{4} + \sigma_6^z\frac{1}{4} + \sigma_1^z\sigma_2^z\frac{1}{4} + \sigma_3^z\sigma_4^z\frac{1}{2} - \sigma_5^z\sigma_6^z\frac{1}{4}$$

Alternatively, given a formula with non-quadratic terms, quadratization [5] can be conducted, resulting in a QUBO (3b) which then can be transformed into H_v (3c). The selection of the route in Fig. 3 is a tradeoff between circuit depth and width, as the PUBO formulation (route 3a) results in deeper circuits, while the QUBO formulation (route 3b and 3c) requires additional ancilla qubits due to introduced auxiliary variables [4].

4.2 Configuration Optimality

The optimality of configurations is encoded in a cost function using feature attributes. For simplicity, we use a cost function searching for minimal feature costs as in Fig. 2, but an arbitrary cost function formulated as a minimization problem - even with multiple attributes for each feature - could be modeled similarly. As in this case, a feature attribute is directly associated with a feature, these can modeled as $\min k(\vec{x}) = \sum_x c_i x_i$ (4), where $k(\vec{x})$ calculates the overall configuration cost associated with all selected features. Applied to the example, this yields:

$$k(\vec{x}) = 30x_1 + 20x_2 + 25x_3 + 50x_4 + 10x_5 + 10x_6$$

This QUBO can be transformed into a feature cost Hamiltonian H_{fc} in step (5) using the transformation explained in step (3), yielding the following Hamiltonian for the running example:

$$H_{fc} = 72.5 * I - 15\sigma_1^z - 10\sigma_2^z - 12.5\sigma_3^z - 25\sigma_4^z - 5\sigma_5^z - 5\sigma_6^z$$

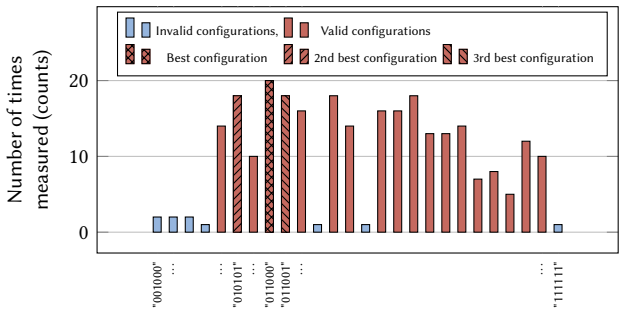


Figure 4: Histogram of executing the QAOA circuit for the example with $\alpha = 200$, $p = 40$ and the final parameters $\vec{\beta}_{end}$ and $\vec{\gamma}_{end}$ for a total of 256 times.

4.3 Combining Validity and Optimality

Finally, to encode the whole problem, the Hamiltonians H_v and H_{fc} are combined using the regulation parameter α which weights the sub-problem Hamiltonians to ensure that the satisfaction as well as the optimization are sufficiently considered in the solution (6). This results in the cost Hamiltonian $H_C = H_{fc} + \alpha H_v$, which we use as phase separating operator U_C to encode the instance-specific information in the QAOA circuit. For example, when choosing $\alpha = 200$ the overall cost results to:

$$H_C = 272.5 * I + 35\sigma_1^z + 40\sigma_2^z - 12.5\sigma_3^z - 25\sigma_4^z - 55\sigma_5^z + 45\sigma_6^z + 50\sigma_1^z\sigma_2^z + 100\sigma_3^z\sigma_4^z - 50\sigma_5^z\sigma_6^z$$

These Hamiltonians can be directly mapped to a quantum circuit by introducing a R_z gate for each single weighted Pauli-Z σ_i^z. Each quadratic or higher term can be decomposed to a single R_z and multiple $CNOT$ gates [8]. The R_z gates are parameterized by γ and weighted by the factor of the corresponding σ_i^z.

4.4 Executing QAOA

With this, we can construct a quantum circuit for the phase separating operator U_C, and by extension also for the whole QAOA quantum routine (see Fig. 1 already with a concrete mixing operator U_M). This allows the execution of QAOA for a specific instance of an attributed feature model. After successfully executing QAOA, a final set of parameters $\vec{\beta}_{end}$ and $\vec{\gamma}_{end}$ is obtained on which the classical optimizer converged. Now, the QAOA circuit can be executed multiple times with these parameters to obtain a histogram of the number of times each configuration (counts) was measured (see Fig. 4). In the histogram, only measured configurations encoded as bitstrings are depicted. The valid configurations are colored in red and invalid configurations in blue. One can see that all 18 valid configurations of the running example are present in the example histogram. Due to the nature of using an approximation algorithm, also invalid configurations were measured with a low probability. The solution to the configuration selection problem "011000" was obtained most frequently in 20 out of the 256 measurements. Similarly, regarding configuration prioritization, when searching m configurations, one can choose the m configurations in the histogram with the highest counts. In case multiple configurations have the same count, we use lexicographic rank. For the example with $m = 3$, the three best configurations are highlighted in the plot.

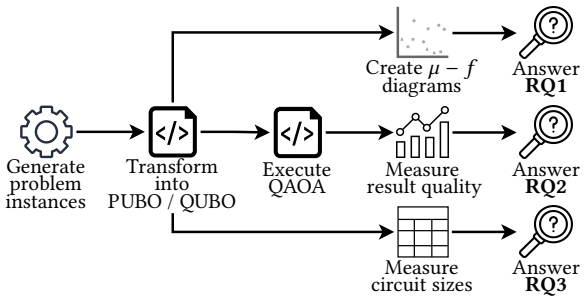


Figure 5: Evaluation methodology.

5 EVALUATION

For evaluation, we pose research questions, develop and conduct experiments, discuss results, and explain threats to validity.

5.1 Research Questions

We pose the following research questions to evaluate our approach:

- **RQ1** Can QAOA solve configuration selection and prioritization? (feasibility)
- **RQ2** What is the quality of results produced by QAOA for small problem instances? (solution quality)
- **RQ3** How do quantum hardware requirements scale with problem instance size? (scalability)

First, with **RQ1**, we evaluate if our approach is suitable for our configuration problems. Second, with **RQ2**, we evaluate if the approach can obtain sufficiently good solutions. We consider small problem instances due to current hardware and simulation limitations. Last, **RQ3** provides insights into the scaling in terms of quantum resources required for our approach.

5.2 Methodology

Our evaluation methodology is depicted in Fig. 5. A set of problem instances is required to conduct the evaluation. As attributed feature models of suitable size are not readily available in the literature, we chose to generate them. For this, we generate 20 feature models (5 with 6, 11, 16, and 21 features each) using the tool FeatureIDE [20] to ensure the well-structuredness of the generated models. The first four columns of Tab. 1 depict information on these problem instances. Feature costs are generated as random integers from the interval [10, 100]. For further evaluation, the generated feature models are transformed into the corresponding QUBO / PUBO formulation. A Python library[1] was implemented that can be used to translate arbitrary problem instances into the QUBO / PUBO formulations, as well as parameterized quantum circuits. For the parameter α we chose the sum of all feature costs times 1.5, which we empirically determined to be suitable for a set of small instances.

[1] https://github.com/KIT-TVA/qc-configuration-problem

Regarding **RQ1**, the optimization landscape observable in a $\mu - f$ diagram [13] gives insights into the suitability of problem instances for QAOA. A $\mu - f$ diagram structures configurations regarding the energy f of a configuration and the arithmetic mean μ of the difference of energy between a current configuration and their nearest neighbors. The energy f is obtained by evaluating the cost Hamiltonian H_C for the configuration, and the nearest neighbors are configurations with a Hamming distance of 1. A $\mu - f$ diagram indicates the quantitative structure of the optimization landscape, defined by the number, size, and depth of 'valleys' (present at $\mu \geq 0$). A 'thin tail' structure is favorable for local search routines like QAOA (see [13]). The characteristic of these thin tails is a central optimal area that decreases towards the outside in an elongated fashion. To answer **RQ1**, we create and analyze such $\mu - f$ diagrams for the problem instances.

To answer **RQ2**, we execute QAOA with $p = 40$ layers and $\beta = 0.01$, $\gamma = -0.01$ as start parameters for each layer on the generated set of 20 problem instances using the Python library and analyze the results with different metrics. We used the COBYLA optimizer and restricted the maximum number of iterations to 1000 to limit the execution time. For instances 0-4 and instances 6-9, we ran QAOA with both the PUBO cost Hamiltonian and the quadratized QUBO cost Hamiltonian. For instances 5 and 10-19, we only considered PUBO variants, as quadratization introduces too many auxiliary variables to simulate the algorithm in a reasonable amount of time, because execution time for a PUBO variant with 26 variables is estimated to be multiple weeks per instance. We apply the following three metrics to the histograms (similar to Fig. 4) we obtained by running QAOA for the different problem instances. To analyze validity, we introduce the metric *validity quality (VQ)*:

$$VQ := 2^{\#features} * \frac{\sum_{c \in C_v} P(c)}{|C_v|}$$

C_v denotes the set of valid configurations. The probability P over these configurations is accumulated and normalized by the ratio of the number of all possible configurations to the number of valid configurations. VQ measures how much higher the average probability of a valid configuration is using QAOA instead of randomly guessing. To analyze optimality, we introduce *cost quality (CQ)*:

$$CQ(m) := \frac{\sum_{c \in C_v} P(c)}{|C_v|} \Bigg/ \frac{\sum_{m \in C_m} P(m)}{m}$$

$C_m \subseteq C_v$ denotes the set of m best valid configurations. This metric measures how much higher the average probability of one of the best m configurations is by using QAOA instead of randomly guessing from only valid configurations. To answer if QAOA can be used to solve configuration prioritization, we evaluate the results using rank-biased overlap (*RBO*) [22, Eq. 23] to compare the list of valid configurations sorted by cost with an equally long list of configurations sorted in descending order of probability. *RBO* measures the similarity of two lists, with the weight being higher at the head of the lists. How quickly the weight decreases is determined by the parameter $p \in 0, 1$, with faster decrease for lower values of p [22].

Finally, for **RQ3**, we analyze the quantum circuits created from the problem instances in the corresponding QUBO / PUBO formulation. To get insights into scalability, we measure the width and depth of the problem circuit resulting from the phase separating operator U_C per layer, which in our QAOA variant has the most impact on scaling. Circuit width refers to the number of qubits the circuit has and circuit depth refers to the maximal number of operations on qubits from input until output.

Table 1: Problem instance sizes and corresponding quantum circuit depth and width of the phase separating operator U_C

ID	# features	# clauses	Max # literals	Depth		Width	
				PUBO	QUBO	PUBO	QUBO
0	6	7	6	300	67	6	15
1	6	6	2	17	17	6	6
2	6	9	2	11	11	6	6
3	6	8	2	15	15	6	6
4	6	8	2	15	15	6	6
5	11	20	7	813	123	11	29
6	11	18	2	24	24	11	11
7	11	16	6	314	70	11	20
8	11	15	2	26	26	11	11
9	11	21	4	92	53	11	14
10	16	18	6	314	81	16	25
11	16	18	7	744	156	16	32
12	16	20	4	82	52	16	18
13	16	21	6	327	82	16	26
14	16	24	2	33	33	16	16
15	21	29	2	38	38	21	21
16	21	37	4	85	68	21	27
17	21	36	5	149	65	21	25
18	21	26	3	40	40	21	21
19	21	30	6	328	82	21	32

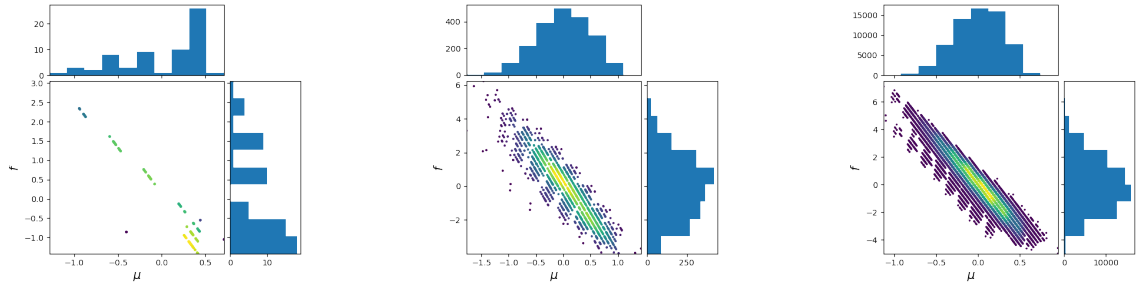
5.3 Results and Discussion

Regarding **RQ1**, $\mu - f$ diagrams for the problem instances were created. Fig. 6 exemplarily depicts the $\mu - f$ diagrams for 3 instances with 6, 11, and 16 features respectively. The optimization landscapes of our problem instances with more than 6 features show a thin tail structure, indicating to be favorable for QAOA. We suppose that the instances with 6 features are too small to show this trend.

RQ1: Can QAOA solve configuration selection and prioritization? (feasibility)

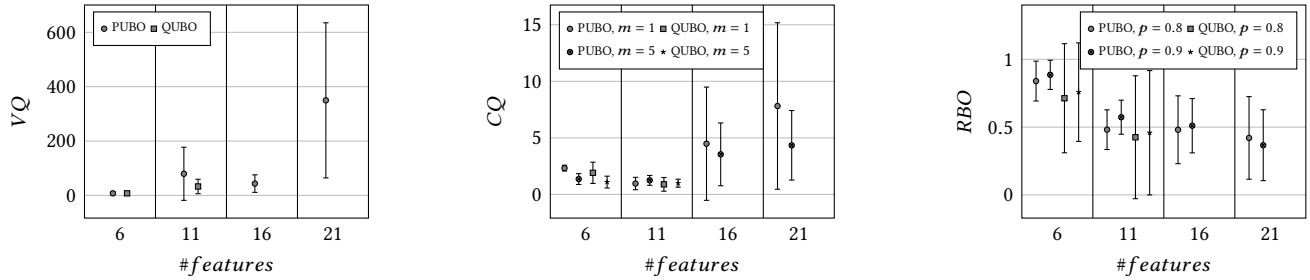
The optimization landscapes of our problem instances with more than 6 features show a thin tail structure in a $\mu - f$ diagram, indicating to be favorable for QAOA.

Fig. 7 visualizes the results obtained for the metrics VQ, CQ, and *RBO*. We observe that the validity quality VQ (see Fig. 7a) increases for bigger problem sizes, with an average of 300 times higher probability to measure a valid configuration over an invalid configuration for instances with 21 features using the PUBO Hamiltonian. We assume this trend results from the restrictive nature of feature models, in which the amount of valid configurations increases less than the amount of all possible configurations on average when increasing the size of feature models. The average cost quality CQ (see Fig. 7b) also increases for larger problem instances. We also observe CQ decreasing for larger m, indicating that more optimal configurations are obtained with a higher probability than less optimal ones. For *RBO* (see Fig. 7c), we observe a downward trend, indicating that the similarity at the beginning of the sequence of measured configurations and the optimal sequence decreases for a higher number of features. However, as we observed for CQ, the average



(a) Instance 0 with 6 features (b) Instance 5 with 11 features (c) Instance 10 with 16 features

Figure 6: $\mu - f$ diagrams [13] for 3 problem instances.



(a) Validity quality VQ. (b) Cost quality CQ for $m = 1$ and $m = 5$. (c) RBO [22] for $p = 0.8$ and $p = 0.9$.

Figure 7: Average result quality of VQ, CQ, and RBO for benchmark instances. Error bars indicate standard deviation. For all metrics, a higher value indicates better result quality.

probability tends to increase for configurations with lower costs. Thus, we suspect, that sorting only the beginning of the sequence of measured configurations could increase the *RBO* value significantly. In general, we observed that the quality of results produced by QAOA is highly dependent on the specific instance, causing a high standard deviation. In all the plots, the QUBO variant performs slightly worse than the PUBO variant. Also, the deviation regarding the *RBO* metric increased when using QUBO. Nevertheless, as we used an ideal simulator, we didn't consider noisy quantum gates, which might favor the QUBO approach requiring fewer gates.

RQ2: What is the quality of results produced by running QAOA for small problem instances? (solution quality)

QAOA shows good results regarding configuration selection, as the validity quality VQ and the cost quality CQ increase with problem instance size. However, regarding configuration prioritization, we observed a decrease in *RBO* with increasing problem instance size.

Measured circuit width and depth of the phase separating operator U_C for the problem instances are depicted in the last four columns of Tab. 1. In general, we can observe, that for the QUBO variant, the width increases and the depth decreases with a higher number of literals per clause. Depth also increases with a higher number of clauses. As the PUBO variant does not require any auxiliary variables, the circuit width is equal to the number of features. The depth depends on the number of clauses and the number of literals per clause. The exact circuit sizes vary with the structure of the specific instance.

Nonetheless, we can determine an upper limit for the number of gates used. As the circuit complexity for the QUBO variant highly depends on the method used for quadratization [5] we provide the limit for the PUBO variant. For a clause with k_i different variables, we can get 1 to 2^{k_i} different summands when transforming a CNF formula into a PUBO depending on how many variables are negated. In the worst case, after the tranformation to the Ising model, we are left with a term containing $\sum_{j=0}^{k_i} \binom{k_i}{j} = 2^{k_i}$ different summands. This means for a CNF formula with m clauses, we are left with $O(m \cdot 2^{k_{\max}})$ different summands, with $k_{\max}$ being the maximum number of literals in a clause. To encode this Ising model in the quantum circuit, we need $2(q-1)$ *CNOT* gates and one R_z gate for a summand with q acting qubits and $q \leq k_{\max}$[8]. In total, the problem circuit requires $O(mk_{\max} \cdot 2^{k_{\max}})$ gates per layer. Using gray codes, this number can be reduced further [8].

RQ3: How do quantum hardware requirements scale with problem instances? (scalability)

Our approach requires qubits equal to the number of features using PUBO and additional qubits using QUBO formulation. Circuit depth scales better for QUBO than for PUBO formulation.

5.4 Threats to Validity

5.4.1 Internal threats. The results were obtained using our implementation of the method and algorithm. We performed a code review on our implementation and manually tested it for small examples, but cannot assure that no errors occurred.

5.4.2 External threats. One threat to validity is the choice of the QAOA parameters α, β, γ, and p. The scope of our evaluation was limited due to the runtime overhead introduced by simulating quantum computers. We manually investigated the influence of α on some problem instances to derive an initial recommendation for choosing α. The start β and γ were chosen according to [3] and we analyzed the β-γ landscape [3] which showed that this choice was sufficient for our problem instances. The choice of p, which generally increases the performance of QAOA for bigger values, was also constrained due to simulation overhead. Our evaluation might be biased by the choice of problem instances. We tried to mitigate this threat by generating feature models with varying numbers of features, by generating and investigating 5 models each, and by using FeatureIDE to ensure well-structuredness. Furthermore, the restriction on QAOA iterations to 1000 runs might have decreased result quality but was necessary to limit runtime.

6 RELATED WORK

Our work is positioned between classical analysis for configurable systems and the application of hybrid quantum algorithms. We discuss classical optimization techniques used for product configuration and works applying QAOA to related problems.

A systematic literature review by Ochoa et al. [16] gives an overview of classical optimization techniques used for product configuration. Techniques include constraint programming, evolutionary algorithms, integer linear programming, or fuzzy logic [16]. While constraint programming and evolutionary algorithms were identified as the most commonly used techniques, the latter showed better performance and scalability [16]. Prioritization has also been researched concerning the testing of product lines [1, 11, 17]. To the best of our knowledge, product configuration problems have not been addressed with quantum computing before.

Blekos et al. [2] give an overview of existing applications of QAOA. Boulebnane and Montanaro thoroughly investigate the usage of QAOA for SAT [3]. Further applications are, e.g., on Max-Cut [21] and Max-3-SAT [23] which also transform a QUBO/PUBO into the Ising formulation. To the best of our knowledge, no work addressed a combination of SAT and literal costs using QAOA.

7 CONCLUSION

We proposed a method to transform the configuration selection and prioritization problems encoded in attributed feature models into a quantum mechanical formulation suitable for optimization problems. The method is evaluated for 20 generated problem instances, showing that QAOA obtains good results regarding configuration selection, but for configuration prioritization, the approach needs further improvement. Fine-tuning QAOA with different mixing operators or classical optimizers, or warm-starting the algorithm could improve it. Also, the influence and selection of our methods regulation parameter α has to be investigated further.

REFERENCES

[1] Al-Hajjaji, M., Lity, S., Lachmann, R., Thum, T., Schaefer, I., and Saake, G. Delta-Oriented Product Prioritization for Similarity-Based Product-Line Testing. In *2017 IEEE/ACM 2nd International Workshop on Variability and Complexity in Software Design (VACE)* (Buenos Aires, Argentina, May 2017), IEEE, pp. 34–40.

[2] Blekos, K., Brand, D., Ceschini, A., Chou, C.-H., Li, R.-H., Pandya, K., and Summer, A. A Review on Quantum Approximate Optimization Algorithm and its Variants, June 2023.

[3] Boulebnane, S., and Montanaro, A. Solving boolean satisfiability problems with the quantum approximate optimization algorithm, Aug. 2022.

[4] Bybee, C., Kleyko, D., Nikonov, D. E., Khosrowshahi, A., Olshausen, B. A., and Sommer, F. T. Efficient optimization with higher-order ising machines. *Nature Communications 14* (Sept. 2023).

[5] Dattani, N. Quadratization in discrete optimization and quantum mechanics, Sept. 2019.

[6] Eichhorn, D., Pett, T., Osborne, T., and Schaefer, I. Quantum computing for feature model analysis: Potentials and challenges. In *Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume A* (New York, NY, USA, 2023), SPLC '23, Association for Computing Machinery, p. 1–7.

[7] Farhi, E., Goldstone, J., and Gutmann, S. A Quantum Approximate Optimization Algorithm, Nov. 2014.

[8] Glos, A., Krawiec, A., and Zimborás, Z. Space-efficient binary optimization for variational quantum computing. *npj Quantum Information 8*, 1 (2022), 39.

[9] Glover, F., Kochenberger, G., Hennig, R., and Du, Y. Quantum bridge analytics I: A tutorial on formulating and using QUBO models. *Annals of Operations Research 314*, 1 (July 2022), 141–183.

[10] Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing - STOC '96* (Philadelphia, Pennsylvania, United States, 1996), ACM Press, pp. 212–219.

[11] Henard, C., Papadakis, M., Perrouin, G., Klein, J., Heymans, P., and Le Traon, Y. Bypassing the Combinatorial Explosion: Using Similarity to Generate and Prioritize T-Wise Test Configurations for Software Product Lines. *IEEE Transactions on Software Engineering 40*, 7 (July 2014), 650–670.

[12] Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, A. Feature-oriented domain analysis (foda) feasibility study. Tech. Rep. CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1990.

[13] Kossmann, G., Binkowski, L., van Luijk, L., Ziegler, T., and Schwonnek, R. Deep-Circuit QAOA, Feb. 2023.

[14] Lucas, A. Ising formulations of many NP problems. *Frontiers in Physics 2* (2014).

[15] Nielsen, M. A., and Chuang, I. L. *Quantum Computation and Quantum Information*, 10th anniversary ed ed. Cambridge University Press, Cambridge ; New York, 2010.

[16] Ochoa, L., González-Rojas, O., Juliana, A. P., Castro, H., and Saake, G. A systematic literature review on the semi-automatic configuration of extended product lines. *Journal of Systems and Software 144* (Oct. 2018), 511–532.

[17] Pett, T., Eichhorn, D., and Schaefer, I. Risk-based compatibility analysis in automotive systems engineering. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings* (Virtual Event Canada, Oct. 2020), ACM, pp. 1–10.

[18] Shor, P. W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing 26*, 5 (Oct. 1997), 1484–1509.

[19] Siegmund, N., Sobernig, S., and Apel, S. Attributed variability models: Outside the comfort zone. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (Paderborn Germany, Aug. 2017), ACM, pp. 268–278.

[20] Thüm, T., Kästner, C., Benduhn, F., Meinicke, J., Saake, G., and Leich, T. Featureide: An extensible framework for feature-oriented software development. *Sci. Comput. Program. 79* (2014), 70–85.

[21] Wang, Z., Hadfield, S., Jiang, Z., and Rieffel, E. G. Quantum Approximate Optimization Algorithm for MaxCut: A Fermionic View. *Physical Review A 97*, 2 (Feb. 2018), 022304.

[22] Webber, W., Moffat, A., and Zobel, J. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS) 28*, 4 (2010), 1–38.

[23] Yu, Y., Cao, C., Wang, X.-B., Shannon, N., and Joynt, R. Solution of SAT problems with the adaptive-bias quantum approximate optimization algorithm. *Physical Review Research 5*, 2 (June 2023), 023147.

ACKNOWLEDGMENTS

This work has been supported by the Ministry of Economic Affairs, Labor and Tourism Baden-Wuerttemberg in the project SEQUOIA End-to-End under Grant No.: WM3-4332149/44. and by the German Ministry for Education and Research in project QuBRA under Grant No.: 13N16303.