Koch et al. *EPJ Quantum Technology* (2026) 13:59
https://doi.org/10.1140/epjqt/s40507-026-00526-7

EPJ.org

EPJ Quantum Technology

RESEARCH Open Access

Resource-efficient quantum optimization via higher-order encoding

Frederik Koch[1*], Shahram Panahiyan[1,2], Rick Mukherjee[1,3,4,5], Joseph Doetsch[6] and Dieter Jaksch[1,7,8]

*Correspondence:
frederik.koch@uni-hamburg.de
[1]Institute for Quantum Physics, University of Hamburg, Luruper Chaussee 149, 22761, Hamburg, Germany
Full list of author information is available at the end of the article

Abstract

Quantum approaches to combinatorial optimization problems (COPs) are often limited by the resource demands of Quadratic Unconstrained Binary Optimization (QUBO) encodings, which enlarge circuits through penalty terms and increase qubit and gate counts. We show that Higher-Order Unconstrained Binary Optimization (HUBO) enables a more resource-efficient formulation. Our method systematically constructs HUBO Hamiltonians and, compared to a QUBO formulation in benchmarks on Gate Assignment (GAP), Maximum k-Colorable Subgraph (MkCS), and Integer Programming (IP) problems, significantly reduces qubit requirements and decreases total CNOT gate counts by at least 89.6% for all tested instances. These results highlight HUBO as a practical alternative for quantum optimization on near-term devices. To promote adoption, we release an open-source Python library that automates HUBO model construction, extends beyond the examples presented in this work, and broadens access to resource-efficient quantum optimization.

Keywords: Quadratic Unconstrained Binary Optimization (QUBO); Higher-Order Unconstrained Binary Optimization (HUBO); Polynomial Unconstrained Binary Optimization (PUBO); Quantum Approximate Optimization Algorithm (QAOA); Combinatorial Optimization Problems (COPs); Graph Coloring; Gate Assignment Problem (GAP); Integer Programming (IP); Quantum Optimization (QO); Quantum Circuit (QC)

1 Introduction

Combinatorial optimization problems are prominent in numerous scientific and industrial domains, ranging from logistics and finance [1] to computational biology [2] and operations research [3]. Challenging problems require finding (near)-optimal solutions within vast discrete configuration spaces, where the search complexity often grows exponentially with problem size [4]. Theoretical insights from statistical physics have illuminated these challenges through their connection to disordered systems, revealing that most industrial optimization problems belong to the NP-hard complexity class [5].

Classical methods, including metaheuristics such as simulated annealing and tabu search [6, 7], and commercial solvers like CPLEX and Gurobi [8, 9] face significant limitations when confronting non-convex energy landscapes. Even relatively small problems (sometimes below 100 variables) remain computationally intractable for them [10–15].

Springer

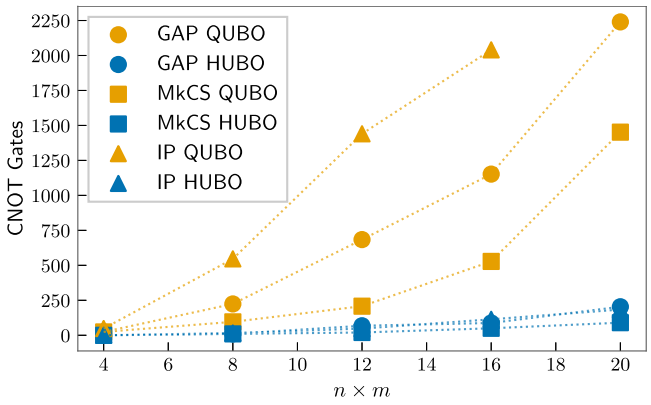

Figure 1 Number of CNOT gates required to reach solution threshold for three COPs, optimized by QAOA with QUBO (orange marker) and HUBO (blue marker) formulations as a function of problem size $n \times m$, with m held fixed. The GAP optimizes the assignment of n aircraft to m airport gates. In the MkCS problem, n vertices are colored with m colors, IP optimizes a polynomial over n variables, each taking one of m possible values. Further details and quantum encodings are provided in the main text

Quantum algorithms, such as adiabatic quantum optimization [16, 17] and the Quantum Approximate Optimization Algorithm (QAOA) [18], a digitized, trotterized variant of adiabatic quantum computing, are poised to address these challenges. This is predominantly done by exploiting mappings of COPs to QUBO models [19–21] which is further facilitated by modeling tools such as Pyqubo [22]. Previous studies [23, 24] have identified several practical limitations of QUBO formulations, including large qubit requirements, costly penalty terms for constraint enforcement, and consequently increased circuit depth with large CNOT gates, which motivate the search for more resource-efficient alternatives.

In contrast, binary encodings, that give rise to HUBO Hamiltonians, directly incorporate such constraints and, therefore, can reduce resource demands. At an intuitive level, each decision variable is represented by a compact binary register, so the single-value assignment is native to the encoding rather than enforced by an additional one-hot penalty term. While demonstrated in specific settings, including protein folding and the traveling salesperson problem [25–29], prior work has lacked a general framework. Moreover, practical concerns remain regarding hardware implementation: higher-order terms formally require multiqubit interactions, whereas present quantum processors primarily support operations involving just one or two qubits at a time [30, 31]. In this work, we use standard decompositions so that such circuits are compiled entirely into single- and two-qubit gates.

By comparing to a QUBO formulation, we demonstrate that the HUBO formulations presented in this work provide substantial and consistent resource advantages for three representative COPs: GAP, MkCS, and IP. These advantages persist after full compilation to native one- and two-qubit gates, with HUBO significantly lowering qubit counts and reducing CNOT requirements by at least 89.6% for all benchmarked problem classes and sizes (see Fig. 1), i.e., without relying on hardware-level higher-order gates. To support broad adoption, we present PyHUBO, an open-source Python library [32], and introduce a systematic framework for constructing HUBO Hamiltonians through binary encodings, showing that for the problems considered, our method yields numerical complexity that

scales polynomially with problem size. Although we focus on QAOA, the approach applies to other algorithms and problems with one-hot or related constraints. Our results demonstrate that HUBO formulations can directly mitigate resource bottlenecks in quantum combinatorial optimization.

2 Methodology

The methodology of this work is structured as follows. In Sect. 2.1, we establish a formal definition and notation for COPs and illustrate this formulation using three representative examples: the GAP, the MkCS, and IP. In Sect. 2.2, we show how the COP formulations from Sect. 2.1 can be systematically mapped to QUBO and HUBO Hamiltonians. Finally, Sect. 2.3 describes the implementation of these Hamiltonians within the QAOA algorithm, utilizing only single- and two-qubit gates.

2.1 Combinatorial optimization problems

A COP consists of a finite set of feasible solutions and an objective function to be optimized [33]. For the remainder of this work, we use the following definition: Consider an n variable COP, where every variable can assume one of m values. Variables are indexed by $1 \leq i \leq n$ and values by $1 \leq v \leq m$. In the following, we will use the terms ith variable and variable i interchangeably to refer to the variable indexed by i, and similarly, we will use the vth value and value v interchangeably to refer to the value indexed by v. A solution $\mathbf{s} \in \{1, \ldots, m\}^n$ assigns a value to each variable. For every $\mathbf{s}$ there exists an objective function $C(\mathbf{s})$ that can be computed in polynomial time and a polynomial time oracle that verifies whether $\mathbf{s}$ is a feasible solution.

The task is to find a solution such that the objective function is minimal among all feasible solutions, i.e., $\mathbf{s}^* = \operatorname{argmin}_{\mathbf{s}} C(\mathbf{s})$. In most applications, the objective function is a polynomial function of the variables [3], and for practical purposes, we express it as

$$C(\mathbf{s}) = \sum_{1 \leq i \leq n} c_1(i, s_i) + \sum_{1 \leq i < j \leq n} c_2(i, j, s_i, s_j) + \cdots, \tag{1}$$

where $c_1(i, s_i)$ is the cost of assigning the value s_i to the ith variable, $c_2(i, j, s_i, s_j)$ the cost of assigning the value s_i to the ith variable while simultaneously assigning the value s_j to the jth variable.

In principle, the objective function can be of any order. However, for most industrial applications, it is linear or quadratic [34], i.e. Eq. (1) only contains terms of the form $c_1(i, s_i)$ or $c_2(i, j, s_i, s_j)$. Genuinely higher-order terms (required for example in [35, 36]) can be incorporated in the same framework; here we restrict to linear and quadratic forms to keep the presentation concise and the scaling analysis transparent.

Often, COPs also involve constraints. For practical purposes, we formulate these constraints as inequalities, which can be converted to equality constraints, e.g., using slack variables [37]. The equality constraints are added as penalty terms to Eq. (1) with appropriately chosen Lagrange multipliers [38].

For example, a COP with a constraint that all variable pairs $(i, j) \in P$ for some set P can not assume the same value can be stated as

$$\begin{aligned} &\underset{\mathbf{s} \in \{1,\ldots,m\}^n}{\text{minimize}} && C(\mathbf{s}) \\ &\underset{\forall (i,j) \in P}{\text{subject to}} && s_i \neq s_j. \end{aligned} \tag{2}$$

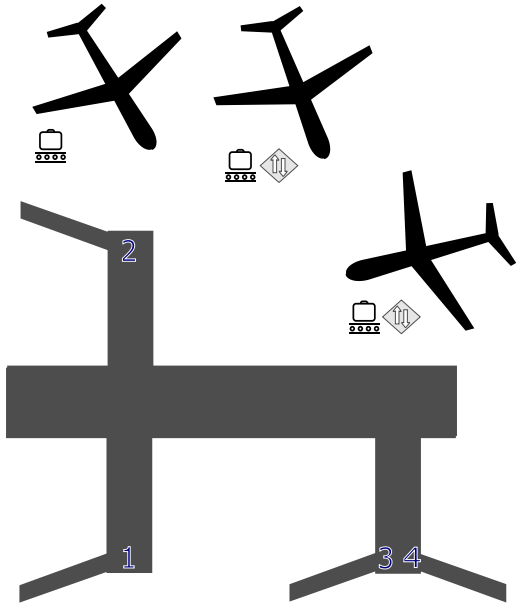

Figure 2 Illustration of the GAP with $m = 4$ airport-gates and $n = 3$ flights. Two of the flights have passengers who require a transfer connection, which is represented by the two-way symbol. A third flight has only arriving and departing passengers, illustrated by the luggage symbol. The goal is to assign airport-gates to flight such that the total passenger walking time is minimized

The penalty term for this constraint has the form

$$\begin{aligned} &c_2(i,j,v,v) = \lambda \\ &\forall (i,j) \in P \quad \forall v \in \{1,\ldots,m\}, \end{aligned} \tag{3}$$

with the Lagrange multiplier λ chosen large enough such that the optimal solution respects the constraint. By adding this penalty term to the objective function, we can transform the COP into an unconstrained optimization problem

$$\underset{\mathbf{s}\in\{1,\ldots,m\}^n}{\text{minimize}} \quad C(\mathbf{s}) + \lambda \sum_{(i,j)\in P} \delta_{s_i,s_j}, \tag{4}$$

where δ_{s_i,s_j} is the Kronecker delta function, which is 1 if $s_i = s_j$ and 0 otherwise.

The above notation applies broadly to COPs. We now illustrate this on three representative classes: the GAP, the MkCS problem, and IP. These formulations provide the basis for deriving their QUBO and HUBO encodings.

2.1.1 Gate assignment problem

The GAP is a well-studied challenge in operations research with significant practical relevance to the aviation industry [39]. It involves optimally assigning airport-gates to aircraft to minimize costs, such as passenger walking distances.

Various classical formulations of the GAP are established [39], and the problem has been studied in the context of quantum optimization [28, 40]. We consider the GAP with n flights and m airport-gates, where the objective is to minimize total passenger walking times (see Fig. 2).

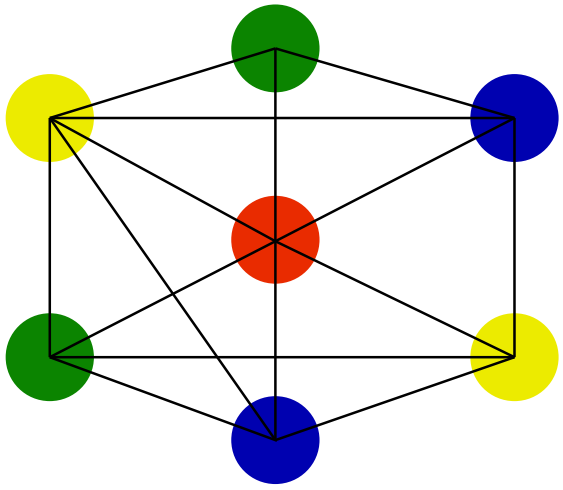

Figure 3 Example instance for the MkCS problem: a properly 4-colored graph with $|V| = 7$. All adjacent vertices have distinct colors, so the objective value (number of monochromatic edges) is zero

We distinguish between two passenger types: those boarding or disembarking, and those transferring between flights. The first type only contributes to the linear coefficients, as the cost of assigning flight i to airport-gate v is given by

$$c_1(i,v) = p_i^{\text{arr}} t_v^{\text{arr}} + p_i^{\text{dep}} t_v^{\text{dep}}, \tag{5}$$

where p_i^{arr} and p_i^{dep} are the numbers of arriving and departing passengers on flight i, and t_v^{arr} and t_v^{dep} are the walking times from the check-in desk to the airport-gate v and from the airport-gate v to the luggage claim, respectively.

The transfer passengers contribute to the quadratic terms through

$$c_2(i,j,v,w) = p_{i,j}^{\text{trans}} t_{v,w}^{\text{trans}}, \tag{6}$$

where $p_{i,j}^{\text{trans}}$ is the number of passengers transferring from flight i to flight j, and $t_{v,w}^{\text{trans}}$ is the walking time between airport-gates v and w.

Finally, we enforce the constraint that temporally overlapping flights cannot share the same airport-gate through a penalty term of the same form as in Eq. (3), where P in the context of the GAP is the set of all conflicting flight pairs.

2.1.2 Maximum k-colorable subgraph problem

The MkCS problem represents a fundamental COP that arises in diverse industrial applications, including network design, frequency assignment, and resource allocation [41–44]. It has attracted considerable attention in both classical and quantum optimization research [45–49].

Given a graph $G = (V, E)$ with vertices V and edges E, the MkCS problem seeks to maximize the size of a properly vertex-colored subgraph with at most k colors (see Fig. 3). A properly vertex-colored subgraph is one where no two adjacent vertices share the same color, and the size of the subgraph is defined as the number of edges in the subgraph.

In the assignment formulation, each vertex is a variable that can assume k values, hence $n = |V|$ and $m = k$. The solution string $\mathbf{s}$ encodes the color of each vertex and s_i is the index for the color assigned to the vertex with index i. The objective function counts the number of edges in the graph that connect vertices of the same color. Minimizing the number of monochromatic edges is equivalent to maximizing the number of edges that connect vertices of different colors, which is the goal of the MkCS problem. Using the notation from Sect. 2.1, the objective function has the coefficients

$$c_2(i,j,v,v) = 1 \quad \forall (i,j) \in E, \tag{7}$$

and all other coefficients are zero.

2.1.3 Integer programming

IP is fundamental to industrial applications such as production planning and resource allocation, as well as a key subject in classical optimization research [50–52]. Recently, quantum and hybrid methods have been employed to address IP problems [20, 53, 54].

Here, we consider an IP problem with n variables and each variable can assume any value from the domain $\{y_1, \ldots, y_m\}$ with $y_i \in \mathbb{Z}$. The task is to assign values to variables such that

$$\underset{\mathbf{u}\in\{y_1,\ldots,y_m\}^n}{\text{minimize}} \quad \mathbf{q}^T\mathbf{u} + \mathbf{u}^T Q\mathbf{u} + \cdots, \tag{8}$$

where $\mathbf{q} \in \mathbb{R}^n$ and $Q \in \mathbb{R}^{n\times n}$ are the coefficients of the objective function. In principle, the objective function Eq. (8) can be of any order. However, for the sake of clarity, here we write out only up to quadratic order. Typically, an IP problem also involves inequality constraints, which often are functions of many variables.

The objective function can be put into the assignment formulation of Sect. 2.1 by writing

$$\begin{aligned} c_1(i,v) &= q_i y_v, \\ c_2(i,j,v,w) &= Q_{ij} y_v y_w, \end{aligned} \tag{9}$$

for all $1 \le v, w \le m$. We find that computing the coefficients of the objective functions under consideration has computational complexity $\mathcal{O}\left(\binom{n}{2}m^2\right)$ (since in general there are $\binom{n}{2}m^2$ quadratic and $n \times m$ linear coefficients).

A range of methods exists to enforce the constraints of IP problems. Penalty-based techniques are most common, encoding constraints as additional terms in the objective function. Examples include slack variable methods that introduce additional variables to convert inequalities into penalizable equalities [37], as well as more recent variants such as unbalanced penalization [24], and Lagrangian approaches [55], both of which avoid increasing the variable space. Alternatively, oracle-based methods verify feasibility directly, without penalties [53]. Together, these strategies are broadly applicable to general constraint enforcement in IP.

In our benchmark examples, we restrict ourselves to constraints that each involve at most two variables. For such simple pairwise constraints, it is most straightforward to explicitly enumerate all violating value pairs, which can be done efficiently with computational complexity $\mathcal{O}(m^2)$ per constraint. Once identified, for each pair of values y_v, y_w

violating a constraint with variables i and j the objective function coefficients are changed

$$c_2(i,j,v,w) \to c_2(i,j,v,w) + \lambda, \tag{10}$$

with a sufficiently large Lagrange multiplier λ. This direct penalty-based formulation is particularly efficient and effective for enforcing simple pairwise constraints in the framework considered here.

2.2 From classical to quantum optimization

The task of mapping a COP to its quantum formulation is twofold. Firstly, we construct a unique mapping between classical solutions and quantum states $\mathbf{s} \leftrightarrow |\mathbf{x}\rangle$. Secondly, we construct a Hamiltonian $\hat{H}_C$ such that $\hat{H}_C |\mathbf{x}\rangle = C(\mathbf{x}) |\mathbf{x}\rangle$. We define $C(\mathbf{x}) \leftrightarrow C(\mathbf{s})$ in which $C(\mathbf{s})$ is given by Eq. (1). Finding the optimal solution to the COP corresponds to finding the ground state of the Hamiltonian. In the following, we discuss two different approaches to this mapping yielding QUBO and HUBO Hamiltonians respectively: the One-Hot and the Binary encoding.

2.2.1 One-hot encoding/QUBO Hamiltonian

In the One-Hot encoding, for each variable value pair (i, v) a binary variable $x_{i,v}$ is created. The ith variable taking the vth value, is encoded by setting the binary variable $x_{i,v} = 1$, while all other binary variables $x_{i,w}$ with $w \neq v$ are set to 0. The objective function Eq. (1) can be written in terms of binary variables $x_{i,v}$ as

$$\begin{aligned} C(\mathbf{x}) = &\sum_{1\le i\le n} \sum_{1\le v\le m} c_1(i,v) x_{i,v} \\ &+ \sum_{1\le i<j\le n} \sum_{1\le v,w\le m} c_2(i,j,v,w) x_{i,v} x_{j,w}. \end{aligned} \tag{11}$$

To ensure that each variable i can only take one value, we must add a so-called one-hot penalty term

$$\lambda \left(1 - \sum_{v=1}^{m} x_{i,v}\right)^2, \tag{12}$$

to the objective function, which ensures that for each variable i exactly one of the binary variables $x_{i,v}$ is 1 and all others are 0. For constrained COPs, other penalty terms need to be added to the cost function to ensure that the constraints are satisfied. For example, the constraint introduced in Sect. 2.1, which requires that all variable pairs $(i,j) \in P$ for some set P cannot assume the same value, is enforced by the penalty term

$$\lambda \sum_{i,j\in P} \sum_{v=1}^{m} x_{i,v} x_{j,v}. \tag{13}$$

Similarly, other constraints can be added as penalty terms to the objective function. A discussion about the construction of the penalty terms can be found in [38].

From the One-Hot formulation, a quantum system is constructed where each One-Hot variable is mapped to one qubit. Then, the value of the qubit $|b\rangle_{i,v}$ with $b \in \{0,1\}$ in the

computational basis represents the value of the One-Hot variable $x_{i,v}$. Since there are n variables and m values per variable, the One-Hot encoding uses exactly $n \times m$ qubits (one qubit for each pair (i, v)). Now the Hamiltonian for this encoding can be found by substituting $x_{i,v} \to (\mathbb{I} - \hat{Z}_{i,v})/2$, where $\hat{Z}_{i,v}$ is the Pauli-Z operator acting on the qubit encoding the variable $x_{i,v}$. This gives the operator form of the QUBO Hamiltonian

$$\begin{aligned}\hat{H}_C = &\sum_{1\le i\le n}\sum_{1\le v\le m} J_{i,v}\hat{Z}_{i,v} \\ &+ \sum_{1\le i,j\le n}\sum_{1\le v,w\le m} J_{i,j,v,w}\hat{Z}_{i,v}\hat{Z}_{j,w},\end{aligned} \tag{14}$$

where the linear cost coefficients $c_1(i, v)$ contribute to the linear QUBO coefficients $J_{i,v}$ and the quadratic cost coefficients $c_2(i, j, v, w)$ contribute to $J_{i,v}$, $J_{j,w}$ and $J_{i,j,v,w}$. It is easy to check that with this Hamiltonian we have $\hat{H}_C |\mathbf{x}\rangle = C(\mathbf{x}) |\mathbf{x}\rangle$. In most numerical studies, the QUBO coefficients $J_{i,v}$ and $J_{i,j,v,w}$ are calculated from $c_1(i, v)$ and $c_2(i, j, v, w)$ through packages such as pyqubo [22, 56].

The One-Hot encoding used here is the most widely adopted encoding that yields a QUBO Hamiltonian. We note, however, that this correspondence is not unique: other encodings (for example, Domain-Wall encoding [23]) can also produce QUBO Hamiltonians. In this work, we focus on One-Hot because of its broad use in the literature. Within this scope we use the terms One-Hot encoding and QUBO formulation interchangeably.

2.2.2 Binary encoding/HUBO Hamiltonian

In the Binary encoding, each variable i is represented using $d = \lceil \log_2(m) \rceil$ bits. To retrieve the value assigned to variable i, we read its associated d bits, convert the binary string to the decimal value v, and conclude that variable i assumes the value $v + 1$ (The +1 is introduced so the bitstring $\vec{0}$ encodes the value with index 1).

For a quantum system with $n \times d$ qubits encoding the COP solution, measuring the d-qubit state $|\mathbf{b}\rangle_i \equiv |b_1\rangle_{i,1} \cdots |b_d\rangle_{i,d}$ in the computational basis yields a bitstring that, when converted to a decimal v, sets the variable with index i to the value with index $v + 1$. While a similar encoding of integer values has been discussed in [23], here we use the encoded integers to denote value indices. This allows us to encode values beyond the set of integers, e.g., colors or gates. In this encoding, the objective function can be written as

$$\begin{aligned}\hat{H}_C = &\sum_{1\le i\le n}\sum_{1\le v\le m} c_1(i, v) |v\rangle \langle v|_i \\ &+ \sum_{1\le i<j\le n}\sum_{1\le v,w\le m} c_2(i, j, v, w) |v\rangle \langle v|_i \otimes |w\rangle \langle w|_j .\end{aligned} \tag{15}$$

It is straightforward to check that we have $\hat{H}_C |\boldsymbol{x}\rangle = C(\boldsymbol{x}) |\boldsymbol{x}\rangle$ with this encoding.

Furthermore, if m is not a power of two, we also add a penalty term in the form of

$$\lambda \sum_{i=1}^{n}\sum_{v=m}^{2^d} |v\rangle \langle v|_i , \tag{16}$$

to penalize all states that encode indices larger than m. Additionally, to incorporate the penalty term (3), we can add a term of the form

$$\lambda \sum_{i,j \in P} \sum_{\nu=1}^{m} |\nu\rangle \langle\nu|_i \otimes |\nu\rangle \langle\nu|_j , \tag{17}$$

to the Hamiltonian. The addition of a one-hot constraint (Eq. (12)) is not required in this encoding and, as we will show, this substantially reduces the resource requirements.

For the construction of the quantum circuit, however, we need to encode the Hamiltonian in terms of Pauli matrices. This can be done by noting that each projector-term can be written out as

$$\begin{aligned} |\nu\rangle \langle\nu|_i &= |\nu_1\rangle \langle\nu_1|_{i,1} \cdots |\nu_d\rangle \langle\nu_d|_{i,d} \\ &= \frac{\mathbb{I}_{i,1} + (-1)^{\nu_1} \hat{Z}_{i,1}}{2} \cdots \frac{\mathbb{I}_{i,d} + (-1)^{\nu_d} \hat{Z}_{i,d}}{2} \\ &= \frac{1}{2^d} \sum_{S \subseteq \{1,\ldots,d\}} \prod_{a \in S} (-1)^{\nu_a} \hat{Z}_{i,a}, \end{aligned} \tag{18}$$

where ν_a is the ath bit of the binary representation of ν and $\hat{Z}_{i,a}$ is the Pauli-Z operator acting on the qubit encoding the ath bit of the ith variable. Note that the sum over S runs over all subsets of the set $B = \{1, \ldots, d\}$, which includes the empty set. The empty set contributes a global factor of $\mathbb{I}/2^d$ to the sum.

To simplify the notation, we define the product of Pauli-Z operators for a variable i and a set $S \subseteq B$ as

$$\hat{Z}_{i,S} = \prod_{a \in S} \hat{Z}_{i,a}. \tag{19}$$

Additionally, for a set S and a value ν, we define

$$R_S(\nu) = \frac{1}{2^d} \prod_{a \in S} (-1)^{\nu_a}. \tag{20}$$

This allows us to express the projector in a more compact form as

$$|\nu\rangle \langle\nu|_i = \sum_{S \subseteq B} R_S(\nu) \hat{Z}_{i,S}. \tag{21}$$

Substituting (21) into the Hamiltonian (15) yields

$$\begin{aligned} \hat{H}_C = &\sum_{1 \le i \le n} \sum_{S \subseteq B} J_{i,S} \hat{Z}_{i,S} \\ &+ \sum_{1 \le i < j \le n} \sum_{S_1, S_2 \subseteq B, B} J_{i,j,S_1,S_2} \hat{Z}_{i,S_1} \hat{Z}_{j,S_2}, \end{aligned} \tag{22}$$

where

$$J_{i,S} = \sum_{v=1}^{m} R_S(v) c_1(i,v),$$
$$J_{i,j,S_1,S_2} = \sum_{v,w=1}^{m} R_{S_1}(v) R_{S_2}(w) c_2(i,j,v,w). \tag{23}$$

Equation (22) is a HUBO Hamiltonian with linear, and quadratic terms and terms of order up $2 \times d$ in the Pauli-Z operators. The coefficients $J_{i,S}$ are determined by $c_1(i,v)$ and the coefficients J_{i,jS_1,S_2} by $c_2(i,j,v,w)$. Hence, Eq. (23) provides a systematic way to find the coefficients of the HUBO Hamiltonian from the coefficients of the objective function.

In principle, the HUBO-coefficients $J_{i,S}$ and J_{i,j,S_1,S_2} can be computed from (23). However, this quickly becomes cumbersome from an implementation perspective, requiring careful manual tracking of all coefficients and explicit caching to avoid redundant computations. Instead, we use a technique based on the Walsh-Hadamard transform [57], which we present in Appendix A. We also prove that calculating the HUBO-coefficients has a numerical complexity that scales polynomially with the number of variables n and values m. For example, calculating the HUBO-coefficients for a COP with a quadratic objective function has numerical complexity $\mathcal{O}(n^2 m^4)$. Our open-source repository offers the Python package PyHUBO [32], along with detailed documentation, requirements, and installation instructions, that facilitates the computation of HUBO coefficients for COPs.

2.3 Circuit implementation

In the previous sections, we presented two methods that encode a COP into a Hamiltonian, ensuring that the ground state of the Hamiltonian corresponds to the optimal solution. Various techniques exist to find this ground state, such as adiabatic evolution [16, 58], quantum imaginary time evolution [59–61], and variational methods [62, 63]. In this work, we focus on the QAOA [18], a variational approach well-suited for near-term quantum devices and effective across many COPs [64–69]. QAOA is widely used in the community and has been implemented on multiple quantum devices, making it an ideal benchmark for our encoding schemes [47, 70, 71].

QAOA operates by alternating between two types of unitary operations: the cost unitary $U_C(\gamma) = \exp(-i\gamma \hat{H}_C)$ with the cost Hamiltonian $\hat{H}_C$ as developed in the Sect. 2.2.1 and Sect. 2.2.2, and the mixer unitary $U_M(\beta) = \exp(-i\beta \hat{H}_M)$ (see Fig. 4). Below, we outline the circuit constructions for these unitaries. For a comprehensive overview of QAOA, see [66].

For the diagonal cost unitary, we largely follow the circuit-synthesis framework of Welch et al. for arbitrary diagonal unitaries [72], and apply it here to both QUBO and HUBO Hamiltonians. A key observation that simplifies the circuit construction is that the cost Hamiltonian $\hat{H}_C$ for every encoding, when expressed in the computational basis, is a sum of diagonal terms. This diagonal structure and, equivalently, because all Pauli-Z operators mutually commute, implies that any two summands in the cost Hamiltonian commute with each other. This commutativity property allows us to decompose the cost unitary for both QUBO and HUBO encodings into a product of unitaries that each implement single terms $J_T \hat{Z}_T$ of the Hamiltonians (14) and (22). Here, T is a set of qubit indices J_T is the coefficient for these qubits and, similar to Sect. 2.2.2 we use the notation $\hat{Z}_T = \prod_{t \in T} \hat{Z}_t$.

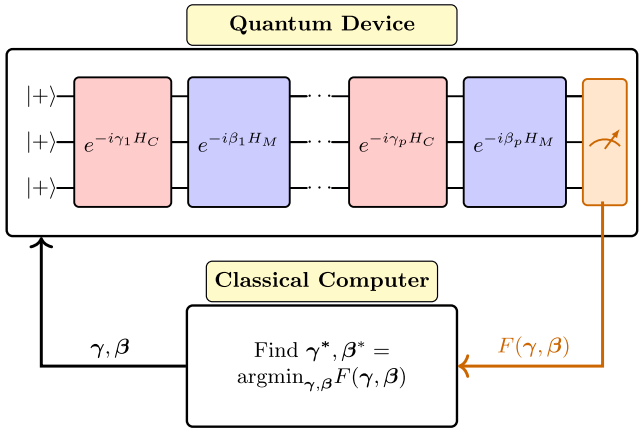

Figure 4 Schematic representation of the QAOA algorithm showing the initialization in the superposition state, the alternating application of cost and mixer unitaries across layers, followed by measurement and classical optimization of variational parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$

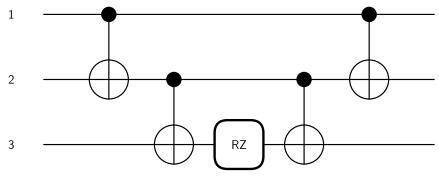

Figure 5 Circuit implementation of the cost Hamiltonian for a three-qubit term, i.e. $\exp(i\gamma J_{1,2,3}\hat{Z}_1\hat{Z}_2\hat{Z}_3)$. The circuit consists of a sequence of CNOT gates to entangle the qubits, such that the initial state $|b_1\rangle_1 |b_2\rangle_2 |b_3\rangle_3$ in the computational basis becomes $|b_1\rangle_1 |b_1 \oplus b_2\rangle_2 |b_1 \oplus b_2 \oplus b_3\rangle_3$ through the first chain of CNOT gates. The single qubit rotation $R_z(2\gamma J_{1,2,3})$ acts on the last qubit. After the rotation, another sequence of CNOT gates disentangles the qubits, resulting in the final state $\exp((-1)^{b_1 \oplus b_2 \oplus b_3} i\gamma J_{1,2,3})$ $|b_1\rangle_1 |b_2\rangle_2 |b_3\rangle_3$

Obviously, in the QUBO encoding, the set T only involves at most two qubit indices, while it can involve up to $2 \times d$ qubit indices in the HUBO encoding.

To construct a circuit for the unitary $\exp(i\gamma J_T \hat{Z}_T)$ using only single- and two-qubit gates, we exploit the fact that the computational basis states $|\mathbf{b}\rangle = \bigotimes_{t\in T} |b_t\rangle_t$, with $b_t \in \{0, 1\}$, form a complete basis. The action of $\exp(i\gamma J_T \hat{Z}_T)$ on such a state results in $\exp\left((-1)^{\oplus_{t\in T} b_t} i\gamma J_T\right) |\mathbf{b}\rangle$. We therefore design a circuit composed of single- and two-qubit gates that reproduces this transformation on computational basis states and thus implements the unitary $\exp(i\gamma J_T \hat{Z}_T)$.

The circuit is constructed in three main steps, as shown in Fig. 5. First, a sequence of $|T| - 1$ CNOT gates is used to transform the initial state $|\mathbf{b}\rangle$ such that one qubit stores the parity $\oplus_{t\in T} b_t$ of the bitstring. Specifically, by arranging the $|T| - 1$ CNOT gates in a chain, where the t-th CNOT acts on the t-th (control) and $(t + 1)$-th (target) qubits of T, the parity is encoded in the last qubit of the set T. Second, a single-qubit rotation $R_z(2\gamma J_T)$ is applied to the parity qubit, generating the phase factor $\exp(-iJ_T\gamma)$ or $\exp(iJ_T\gamma)$ for odd or even parity, respectively. Finally, the sequence of CNOT gates is applied in reverse order to restore the original product state, resulting in $\exp((-1)^{\oplus_{t\in T} b_t} i\gamma J_T) |\mathbf{b}\rangle$. Thus, the circuit implementing the cost unitary for a single $|T|$-th order term in the cost Hamiltonian requires $2(|T| - 1)$ CNOT gates and one R_Z gate.

In principle, each term in (14) and (22) can be implemented in this way sequentially. However, as we show in the Appendix B, constructing the cost layer $U_C(\gamma)$ for all terms of

a HUBO Hamiltonian (22) in this way is not the most resource-efficient approach, and we present an circuit based on the provably optimal Gray-code implementation by Welch et al. [72], which reduces the number of CNOT gates. The key idea is that parity states generated for one term of the Hamiltonian can often be leveraged to efficiently construct the parity states needed for other terms, thereby minimizing redundant operations. Consequently, as we show in Appendix C, the asymptotic CNOT and R_Z gate counts for the Gray-code implementation scale as $O(n^2m^2)$, whereas the sequential implementation scales as $O(n^2m^2\log(m))$.

For the mixer Hamiltonian $\hat{H}_M$, we employ the standard Pauli-X mixer

$$\hat{H}_M = \sum_i \hat{X}_i, \tag{24}$$

which acts on all qubits of the circuit. This choice ensures that the mixer unitary can efficiently explore the computational basis states by creating a superposition that allows transitions between different solution candidates. Furthermore, the mixer unitary can be implemented using a simple layer of single-qubit R_x rotation gates.

To optimize the variational parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, we apply gradient descent methods with the initialization scheme of Zhou et al. [64], which improves convergence and solution quality over random initialization. Gradients of the expectation value with respect to variational parameters are efficiently computed via implicit differentiation [73], avoiding the computational overhead and inaccuracy of finite-difference methods. Quantum circuits and optimization routines are implemented using PennyLane [74], enabling automatic differentiation and GPU acceleration for efficient optimization.

The Lagrange multipliers λ for the different penalty terms in the objective function are chosen iteratively before running the QAOA benchmarking. Only after identifying λ values that ensure the QAOA algorithm consistently finds feasible solutions respecting all constraints do we proceed to run and evaluate the algorithm's performance using these selected penalty parameters. This iterative approach ensures that the penalty terms are effective in guiding the QAOA algorithm towards feasible solutions while still allowing for exploration of the solution space.

The quality of QAOA solutions is typically evaluated via the expectation value of the cost Hamiltonian. For COPs, however, feasibility must also be considered. Hence, we employ a metric similar to the one introduced in [75], here referred to as the approximation ratio, defined as

$$A = 1 - \sum_{\mathbf{b}\in B_{\text{feas}}} r(\mathbf{b}) |\langle \psi(\boldsymbol{\gamma},\boldsymbol{\beta})| |\mathbf{b}\rangle|^2, \tag{25}$$

where $\{|\mathbf{b}\rangle \,|\mathbf{b} \in B_{\text{feas}}\}$ is the set of computational basis states satisfying all constraints and

$$r(\mathbf{b}) = \frac{C_{\max} - C(\mathbf{b})}{C_{\max} - C_{\min}}, \tag{26}$$

rescales $C(\mathbf{b})$ to the interval $[0, 1]$. Here, $C_{\min}$ and $C_{\max}$ are the minimal and maximal values of the objective function, determined using commercial solvers for the benchmark instances. The approximation ratio is particularly suitable for our benchmarks, as it immediately accounts for the treatment of invalid solutions as equivalent to maximum cost

Table 1 Exact qubit requirements and analytical scaling for CNOT and R_Z gates per QAOA layer in both encodings, where n is the number of variables and m the number of values per variable. These scalings are based on worst case assumptions (see Appendix C). In practice, gate counts are often lower (see Table 2)

	# Qubits	# CNOTs	# R_Z
QUBO	nm	$O(n^2m^2)$	$O(n^2m^2)$
HUBO	$n\lceil log_2(m)\rceil$	$O(n^2m^2)$	$O(n^2m^2)$

Table 2 Quantum Resources required per QAOA layer to encode the GAP instance introduced in Fig. 6, an MkCS instance illustrated in Fig. 9 and an IP instance shown in Fig. 12

	# Qubits	# CNOTS	# R_Z
GAP-QUBO	20	140	90
GAP-HUBO	10	68	27
MkCS-QUBO	20	132	86
MkCS-HUBO	10	90	27
IP-QUBO	16	120	76
IP-HUBO	8	38	31

states. This metric combines solution utility, feasibility, and objective value, and can be applied to all three COPs discussed in the results section.

If the quantum state $|\psi(\boldsymbol{\gamma},\boldsymbol{\beta})\rangle$ corresponds to the optimal solution, then $A = 0$. If it yields only infeasible or maximal-cost solutions, $A = 1$. Thus, A can be interpreted as the expectation value of a normalized cost Hamiltonian, where feasible eigenstates have eigenvalues $[0, 1]$, while infeasible eigenstates can only have eigenvalue 1. Lower values of A indicate higher-quality solutions found by QAOA.

Since the set of feasible bitstrings B_{feas} grows exponentially with problem size, evaluation of the approximation ratio becomes numerically taxing even for modestly sized COPs. To address this, we approximate Eq. (25) using a sampling-based method. Further details on this approach and other numerical techniques used for benchmarking are provided in Appendix D.

3 Results

To evaluate resource efficiency and solution quality, we analyze three complementary metrics across all benchmarks: (i) the number of qubits required to represent the problem, (ii) the gate count per QAOA layer after circuit compilation (i.e. # CNOTS, # R_Z), and (iii) the total number of gates needed to achieve target approximation ratios (A).

For qubit requirements, a COP with n variables each taking m values requires $n \times m$ qubits under QUBO, but only $n \times \lceil \log_2(m) \rceil$ under HUBO. Both encodings grow linearly in n, however, the growth in m is linear for QUBO but logarithmic for HUBO, yielding exponential savings when variables admit many values. This advantage is particularly critical in problems such as the GAP, where each flight-variable can assume a large number of airport-gate-values [39].

For gate counts per QAOA layer, we compile the cost unitary circuits to contain only rotation and CNOT gates, and report those as our primary resources. Both encodings scale quadratically in gate count (see Table 1). In practice, HUBO consistently compiles to fewer gates (Table 2), often by large margins, for instance, a 69% per-layer reduction in CNOT gates for the considered IP instance. These savings can largely be attributed to the penalty overhead from the QUBO encoding (for details see Appendix D.2). It should

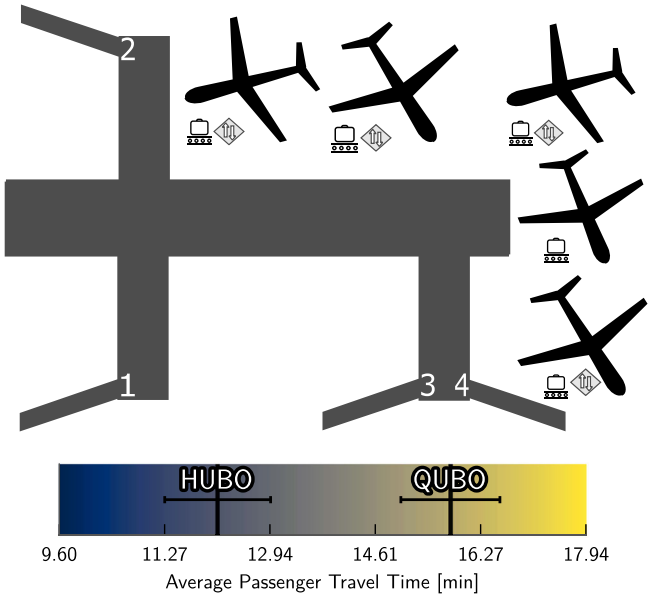

Figure 6 Upper panel: Airport-Gate layout for the benchmark instance with 4 airport-gates and 5 flights. Lower panel: The range of feasible objective values for total passenger time is shown, including the minimum (blue) and maximum (yellow) obtained by a commercial solver. The average objective values achieved by the QAOA algorithm with QUBO and HUBO encodings (10 layers) are shown for comparison

be noted that Hadamard gates for state initialization and R_X gates for the mixer are not reported separately since their counts equal the number of qubits.

Finally, while increasing the number of QAOA layers is known to improve solution quality [18], feasibility on real hardware depends on total gate counts rather than per-layer counts alone. In what follows, we will show that both the per-layer gate requirements and the convergence rate toward the optimum vary significantly with the chosen encoding. Since no analytical bounds exist for the minimum depth needed to reach a given solution threshold, we rely on empirical benchmarking. Here, the most notable advantage of HUBO over QUBO is evident, with HUBO achieving savings of up to 97.1% in CNOT counts for IP benchmarks. In the following subsections, we therefore compare QUBO and HUBO across three representative COPs, reporting solution quality at fixed depths, convergence behavior as a function of gate count, and total resources needed to achieve specified solution thresholds. All averages are over 100 independently optimized QAOA parameters, with error bars representing the standard deviation, as detailed in Appendix D.

3.1 Gate assignment problem

Using a GAP instance with five flights ($n = 5$) and four airport-gates ($m = 4$), we benchmark both QUBO-QAOA and HUBO-QAOA, with ten layers. Each layer contains gate counts as reported in Table 2. The GAP instance considered has 4 flights with transfer passengers, resulting in 24 nonzero quadratic terms (see Equation (6)) in the classical objective function. Additionally, 3 flights have gate conflicts with 2 other flights each, while 2 flights have gate conflicts with 1 other flight each. The motivation for this scenario, as well as further details such as passenger numbers and walking times, are provided in Appendix E. An illustration of the problem is shown in Fig. 6.

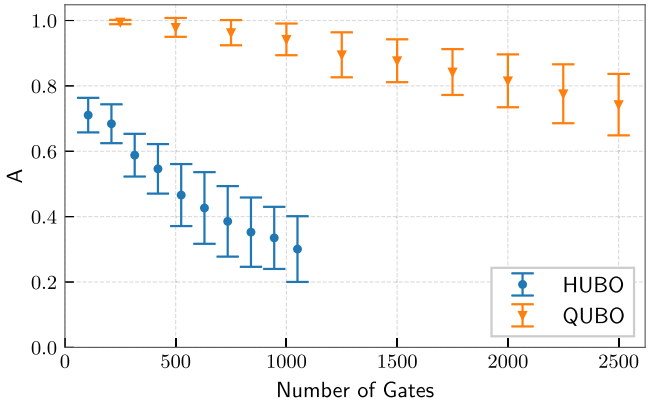

Figure 7 Average approximation ratio (Eq. (25)) versus total gate count for HUBO and QUBO encodings on the 4-airport-gate, 5-flight problem instance. The gate count includes all circuit gates used: CNOT, R_Z, R_X, and Hadamard, the latter applied only once for state initialization (10 for HUBO, 20 for QUBO). For each encoding, the number of gates per QAOA layer is fixed, so increasing the number of layers results in a uniform increment in total gates, producing evenly spaced datapoints along the x-axis. Each point corresponds to a fixed number of QAOA layers, ranging from 1 to 10

The best feasible solution, determined by a commercial solver, yields an average passenger travel time of 9.6 minutes, while the worst feasible assignment totals 17.9 minutes. Within this range of solutions, HUBO-QAOA achieves an average objective value of 12.1 minutes, while QUBO-QAOA's average objective value is 15.8 minutes, demonstrating a substantial solution quality improvement with HUBO for the same number of QAOA layers.

Furthermore, HUBO-QAOA achieves lower approximation ratios using fewer quantum resources. In Fig. 7, we compare the average approximation ratio as a function of total gate count, varying the number of QAOA layers from 1 to 10. Because the same number of gates is added per QAOA layer, the datapoints are evenly spaced along the x-axis. For any fixed number of QAOA layers, the HUBO encoding requires fewer gates in total. More importantly, for any given total gate count, HUBO-QAOA consistently attains much better approximation ratios. For example, with approximately 1040 total gates (1060 for HUBO, 1020 for QUBO), the approximation ratio achieved with QUBO is more than 3 times worse than that achieved with HUBO.

HUBO-QAOA consistently demonstrates resource savings over QUBO-QAOA as the problem size varies. Figure 8 summarizes the number of gates required to achieve an approximation ratio of at least 0.50 for both encodings, as the number of flights increases from 1 to 5 with the number of airport-gates fixed at 4. The smaller GAP instances are generated by repeatedly deleting flight assignments from the instance described in Fig. 6. In this scaling approach, the problem size corresponds directly to the number of qubits required by the QUBO encoding. The figure separately reports the number of CNOT and single-qubit gates, since CNOT gates are typically more experimentally demanding. Across all evaluated problem sizes, the HUBO encoding reduces the number of required CNOT gates between 90.0% and 100% (the single flight GAP instance in the HUBO encoding doesn't require CNOT gates) and the number of R_Z gates between 86.1% and 95.3% compared to the QUBO encoding. These results highlight that substantial quantum resource efficiency gains with the HUBO approach persist across varying problem sizes.

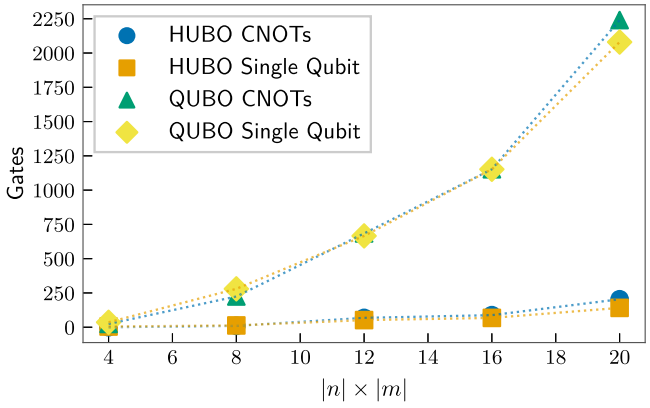

Figure 8 Total number of CNOT and single-qubit gates required to attain an approximation ratio of 0.50, plotted against the problem size of the GAP (number of airport-gates fixed at 4). Problem size is defined as the product of the number of flights and the number of gates and directly corresponds to the qubit count in the QUBO encoding. CNOT and single-qubit gates are shown separately, reflecting their differing experimental costs. Across all problem sizes, the HUBO encoding achieves the target approximation ratio using substantially fewer CNOT and single-qubit gates than the QUBO encoding

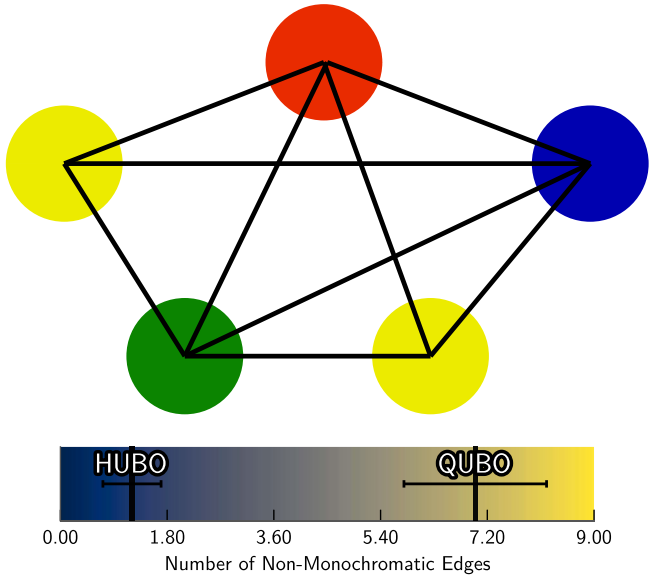

Figure 9 Upper panel: Graph with five vertices and nine edges used as a benchmark instance for the MkCS problem, colored with four colors. The displayed coloring corresponds to the optimal solution with zero monochromatic edges. Lower panel: Range of feasible objective values and average number of monochromatic edges identified by HUBO-QAOA and QUBO-QAOA for a fixed number of layers (3)

3.2 Maximum k-colorable subgraph problem

We benchmark an MkCS problem with $|V| = n = 5$ vertices and $k = m = 4$ colors on the graph illustrated in Fig. 9, using a QAOA circuit with three layers. The objective is to minimize the number of monochromatic edges. For this instance, an optimal coloring results in zero monochromatic edges, while assigning the same color to all vertices yields a maximum objective value of nine. The range of feasible values and the average performance

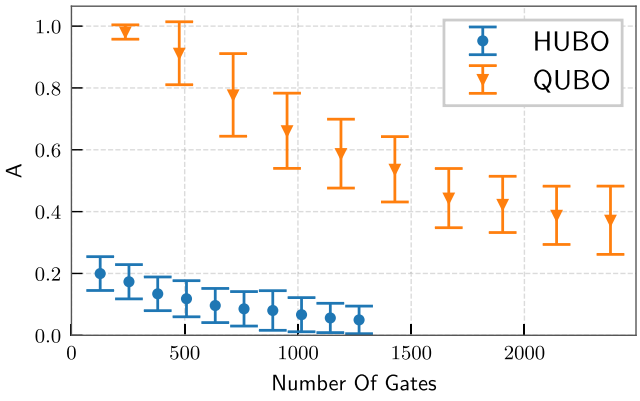

Figure 10 Average approximation ratio achieved by QAOA as a function of total gate count for HUBO and QUBO encodings. Results are shown for the same graph with five vertices and nine edges as in Fig. 9. The number of gates required for each layer can be found in Table 2. Data points are evenly spaced as gate counts scale proportionally with the number of QAOA layers

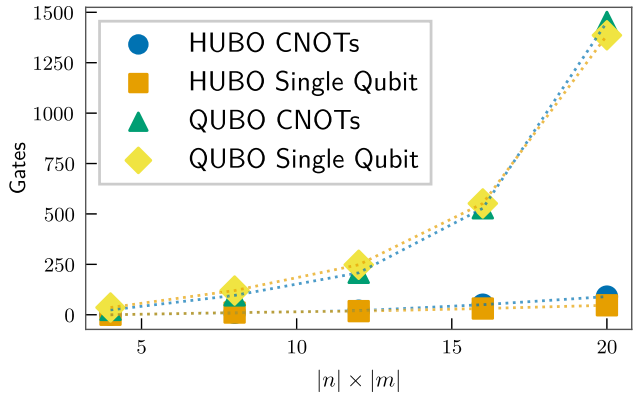

Figure 11 Scaling of quantum resource requirements for MkCS instances. The number of CNOT and R_Z gates required to achieve an approximation ratio of 0.20 is shown as a function of problem size (number of vertices times number of colors, with $m = 4$ fixed). The HUBO encoding consistently requires substantially fewer gates than QUBO across all tested sizes, with the difference increasing for larger instances

of each encoding are shown in the lower panel of Fig. 9. HUBO-QAOA finds, on average, 1.2 monochromatic edges, compared to an average of 6.9 for QUBO-QAOA.

HUBO-QAOA consistently achieves better approximation ratios than QUBO-QAOA while using fewer quantum resources. Figure 10 shows the average approximation ratio obtained by QAOA as a function of total gate count for both encodings. Using the gate numbers reported in Table 2, HUBO-QAOA always reaches approximation ratios below 0.18. In contrast, even with the largest considered number of QAOA layers (10), the QUBO-QAOA algorithm only attains an average approximation ratio of 0.37. Each QUBO-QAOA layer requires 238 gates, compared to only 127 gates for HUBO-QAOA. This substantial difference in resource demands can be attributed to the penalty overhead required by the QUBO encoding.

We find that the quantum resource advantage of HUBO over QUBO persists across all tested MkCS problem sizes. As summarized in Fig. 11, the number of gates required to achieve an approximation ratio of 0.20 increases much more rapidly for QUBO than for

$$f(x_1, x_2, x_3, x_4, x_5) = x_1x_2 - 2x_2x_3 + 1.5x_3x_4 - 0.5x_1x_3 + 3x_3 - 2x_2x_5$$

$$x_1, x_2, x_3, x_4, x_5 \in \{0, 1, 2, 3\}$$

$$3 \geq x_1x_2$$
$$5 \geq x_3 + x_4$$

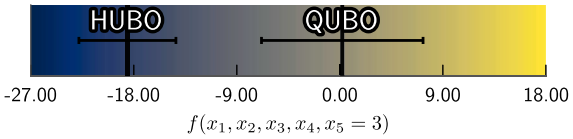

Figure 12 Upper panel: Structure of the IP instance considered, with $n = 5$ variables ($m = 4$ values each), two linear constraints, and both linear and quadratic objective function terms. This instance is referenced here for later HUBO-QAOA benchmarking, as QUBO-QAOA is not computationally feasible for this size. Lower panel: Range of feasible objective values for the reduced IP instance with $x_5 = 3$ fixed. The optimal (–27) and maximal (18) values are shown. Average objective values attained by HUBO-QAOA and QUBO-QAOA after nine layers are indicated for comparison

HUBO as problem size grows. Problem size is defined as the product of the number of vertices and the number of colors in the graph, which matches the qubit requirement for the QUBO encoding. For these benchmarks, the number of colors is fixed at $k = 4$ and the number of vertices varies from 1 to 5. The smaller graphs are generated by deleting vertices from the instance in Fig. 9. CNOT and single-qubit gate requirements are reported separately (note that the one-vertex instance is solved trivially by HUBO and doesn't require any CNOT or RZ gates). Across all evaluated problem sizes, the HUBO encoding reduces the number of required CNOT gates by 89.6 – 100% and the number of R_Z gates by 90.8 – 100% compared to the QUBO encoding.

3.3 Integer programming

To provide context for the subsequent analyses, the upper panel of Fig. 12 illustrates the IP instance considered in this work. This instance consists of $n = 5$ variables, each with $m = 4$ possible values, two linear constraints (each depending on two variables), and an objective function containing both linear and quadratic terms. The minimum feasible objective value is –27, which is achieved by the assignment $x_1 = 0$, $x_2 = 3$, $x_3 = 3$, $x_4 = 0$, and $x_5 = 3$. This instance is used to benchmark HUBO-QAOA; however, QUBO-QAOA was not computationally feasible at this problem size due to excessive computational resource requirements.

To enable a direct comparison between the two encodings, we benchmark QAOA on a reduced version of the problem with $x_5 = 3$ fixed. The lower panel of Fig. 12 displays the range of feasible objective values for this simplified instance, which retains the optimum (–27) and maximum (18) values of the original. For 9 QAOA layers, HUBO-QAOA yields an average objective value of –18.55, while QUBO-QAOA produces an average of 0.21.

Additionally, HUBO-QAOA demonstrates significantly greater resource efficiency and superior approximation ratio compared to QUBO-QAOA across all layer settings, as

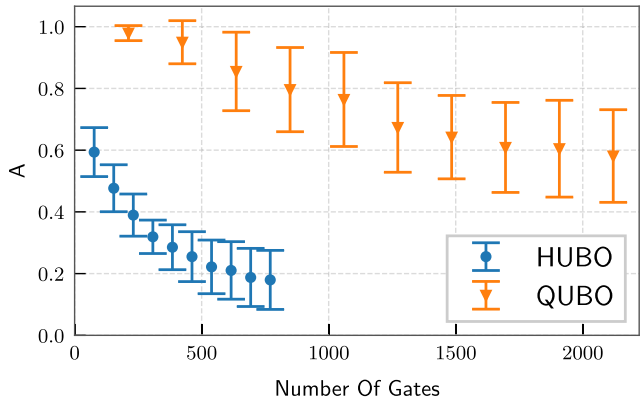

Figure 13 Average approximation ratio versus total gate count for QAOA applied to the reduced IP instance, comparing HUBO and QUBO encodings. HUBO-QAOA consistently achieves higher approximation ratios at lower quantum resource cost than QUBO-QAOA. For example, to reach a target approximation ratio of 0.60, HUBO-QAOA and QUBO-QAOA require 85 and 2136 gates, respectively

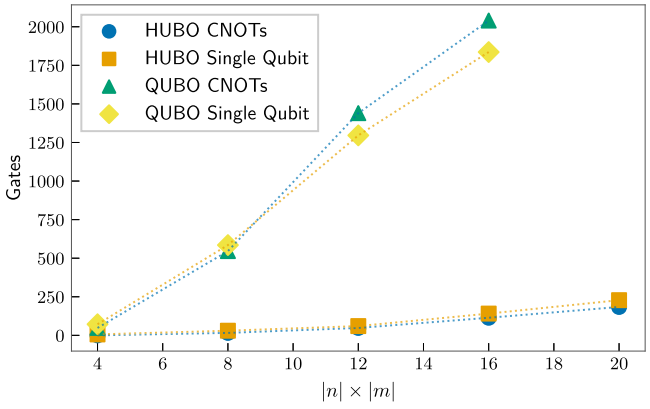

Figure 14 Number of gates required to achieve an approximation ratio of 0.30 as a function of IP problem size, comparing HUBO and QUBO encodings. Both CNOT and single-qubit gate requirements are shown. Problem size is modified by changing the number of variables in the IP problem. HUBO-QAOA consistently achieves the target approximation ratio with dramatically fewer gates than QUBO-QAOA, and remains tractable even for the largest instances where QUBO-QAOA cannot be executed due to computational resource constraints

shown in Fig. 13. The per-layer CNOT and R_Z gate counts for each encoding are provided in Table 2. Evidently, QUBO-QAOA requires over three times more CNOT gates and more than twice as many R_Z gates per layer compared to HUBO-QAOA.

For a targeted approximation threshold, HUBO-QAOA requires substantially fewer gates compared to QUBO-QAOA. For instance, to reach an approximation ratio of 0.60, QUBO-QAOA requires 10 layers, while HUBO-QAOA achieves the same result with only 1 layer. When accounting for both the higher resources per layer and the greater number of layers required, QUBO-QAOA, compared to HUBO-QAOA, consumes 31.6 times more CNOT gates (1200 vs. 38) and 24.5 times more R_Z gates (760 vs. 31).

The resource advantage of HUBO-QAOA over QUBO-QAOA remains robust across all benchmarked IP problem sizes. Figure 14 reports the number of gates required to reach an approximation ratio of 0.30 for different problem sizes, with CNOT and single-qubit gate

requirements shown separately. Problem size is varied by starting from the largest instance (as shown in Fig. 12, with $n = 5$ variables, and $m = 4$ values) and generating progressively smaller instances by fixing variables to their optimal assignments. This approach preserves the minimal objective value while producing a sequence of simpler problems, resulting in increments of 4 qubits per data point.

Across all benchmarked problem sizes, the HUBO encoding reduces CNOT requirements by 94.4 – 100% and the number of R_Z gates by 91.6 – 95.3% compared to QUBO (again, the one variable instance doesn't require any CNOT gates in the HUBO formulation). These results, together with those from earlier figures, demonstrate the persistent resource efficiency advantage of HUBO-QAOA over QUBO-QAOA throughout the tested range of IP problem sizes.

4 Conclusion

We showed that using HUBO formulations can substantially reduce resource bottlenecks that often constrain the practical quantum optimization of COPs. Although one might assume that higher-order interactions would increase quantum hardware demands, our results on three representative problems reveal the opposite. HUBO encoding requires significantly fewer qubits compared to QUBO, and more importantly, leads to substantial reductions in the number of CNOT and single-qubit gates. For each problem class, we observe a reduction of CNOT gates by 89.6 – 100%, which makes experimental realization more feasible for current and near-term quantum devices.

In addition to these resource savings, we have developed a general framework for constructing HUBO models, together with the open-source Python package [32]. This equips the community with a framework to efficiently formulate COPs on quantum hardware, helping to mitigate current resource limitations, and, for quadratic problems, ensures that the Hamiltonian construction scales polynomially with problem size.

Our results suggest that HUBO encodings can be a valuable tool for reducing resource demands in quantum optimization. As quantum technology advances and problem sizes scale up, such resource-efficient methods will be increasingly important for enabling practical applications in industry and research.

Appendix A: Mathematical foundation of HUBO encoding

While in principle it is possible to find the HUBO coefficients using Eq. (23), it quickly becomes cumbersome and numerically expensive to iterate over all required subsets. Instead, we adopt a more efficient, less error-prone, and clearer method to determine the HUBO coefficients. The following derivation is inspired by [57], where the authors show that finding the coefficients of a HUBO Hamiltonian from a Boolean function amounts to Fourier transforming it.

We first note that a single qubit projector can be written out as

$$\begin{pmatrix} c_0 \\ c_1 \end{pmatrix} \begin{pmatrix} |0\rangle\langle 0| \\ |1\rangle\langle 1| \end{pmatrix} = \frac{1}{2} \begin{pmatrix} c_0 + c_1 \\ c_0 - c_1 \end{pmatrix} \begin{pmatrix} \mathbb{I} \\ \hat{Z} \end{pmatrix}. \tag{27}$$

In the following, we will call $\begin{pmatrix} |0\rangle\langle 0| \\ |1\rangle\langle 1| \end{pmatrix}$, p-basis and $\begin{pmatrix} \mathbb{I} \\ \hat{Z} \end{pmatrix}$, j-basis and rewrite Eq. (27) as

$$\begin{pmatrix} c_0 \\ c_1 \end{pmatrix}_p = \frac{1}{2}\begin{pmatrix} c_o + c_1 \\ c_0 - c_1 \end{pmatrix}_j = \hat{H}\begin{pmatrix} c_0 \\ c_1 \end{pmatrix}_j \coloneqq \begin{pmatrix} J_0 \\ J_1 \end{pmatrix}_j, \tag{28}$$

where $\hat{H}$ is the Hadamard matrix providing a transformation from the p-basis to the j-basis.

Generalizing this idea to an d-qubit projector, the coefficient $c_{i_1 i_2 \cdots i_d}$ of the projector $|i_1 i_2 \cdots i_d\rangle\langle i_1 i_2 \cdots i_d|$ in the p-basis is transformed to the j-basis using the Hadamard matrix $\hat{H}^{\otimes d}$, which is the tensor product of d Hadamard matrices. This can be seen by noting that the projector $|i_1 i_2 \cdots i_d\rangle\langle i_1 i_2 \cdots i_d|$ can be written as the tensor product of the single-qubit projectors $|i_1\rangle\langle i_1| \otimes |i_2\rangle\langle i_2| \otimes \cdots \otimes |i_d\rangle\langle i_d|$, each being transformed into the j-basis by a Hadamard matrix, hence the transformation for the entire d-qubit projector is done by the tensor product of Hadamard matrices for all single qubit coefficients.

We use the notation $c_{i_1 i_2 \cdots i_d}$ for the coefficient of the projector $|i_1 i_2 \cdots i_d\rangle\langle i_1 i_2 \cdots i_d|$ in the p-basis and the coefficient $J_{i_1 i_2 \cdots i_d}$ in the j-basis encodes the interaction strength of the HUBO term $\prod_{j=1}^{d} \hat{Z}_j^{i_j}$. Using this notation, we can write the transformation from the p-basis to the j-basis as

$$\begin{pmatrix} c_{0\cdots 00} \\ c_{0\cdots 01} \\ \vdots \\ c_{1\cdots 10} \\ c_{1\cdots 11} \end{pmatrix}_p = \hat{H}^{\otimes d}\begin{pmatrix} c_{0\cdots 00} \\ c_{0\cdots 01} \\ \vdots \\ c_{1\cdots 10} \\ c_{1\cdots 11} \end{pmatrix}_j = \begin{pmatrix} J_{0\cdots 00} \\ J_{0\cdots 01} \\ \vdots \\ J_{1\cdots 10} \\ J_{1\cdots 11} \end{pmatrix}_j. \tag{29}$$

For example, a quadratic COP has the HUBO model

$$\hat{H} = \sum_{1\le i\le n}\sum_{1\le k\le m} c_1(i,k)\,|k\rangle\langle k|_i + \sum_{1\le i<j\le n}\sum_{1\le k,l\le m} c_2(i,j,k,l)\,|k\rangle\langle k|_i \otimes |l\rangle\langle l|_j. \tag{30}$$

We write the linear coefficients for each variable i in a vector $\boldsymbol{c}(i) = (c_1(i,1), c_1(i,2), \ldots, c_1(i,m))^T$ and apply $\hat{H}^{\otimes d}$ to each vector $\boldsymbol{c}_1(i)$ to obtain the coefficients in the j-basis $\boldsymbol{J}_i = \hat{H}^{\otimes d}\boldsymbol{c}_1(i)$. Note that because $2^d = m$, the Hadamard transformation is an $m \times m$ matrix, hence finding the linear coefficients of all n variables in the j-basis has a computational complexity of n matrix multiplications on an m-dimensional vector, hence $O(nm^2)$. Similarly, we can find the quadratic coefficients by applying the Hadamard transform to the vector $\boldsymbol{c}_{ij} = (c_2(i,j,1,1), c_2(i,j,1,2), \ldots, c_2(i,j,m,m))^T$ for each pair of variables (i,j) and obtain the linear coefficients in the j-basis $\boldsymbol{J}_{ij} = \hat{H}^{\otimes d}\boldsymbol{c}_{ij}$. Calculating the quadratic coefficients for all pairs of variables amounts to a computational complexity of n^2 (since there are $\binom{n}{2}$ combinations of variables) and with the vector $\boldsymbol{c}_{i,j}$ being of size m^2 the computational complexity for finding the quadratic coefficients is $O(n^2m^4)$. In general, for a COP of kth order with n variables and m values per variable, the computational complexity for finding the coefficients in the j-basis is $O(\binom{n}{k}m^{2k})$.

This formalism forms the basis of the PyHUBO Python package [32], which is made available as part of this publication.

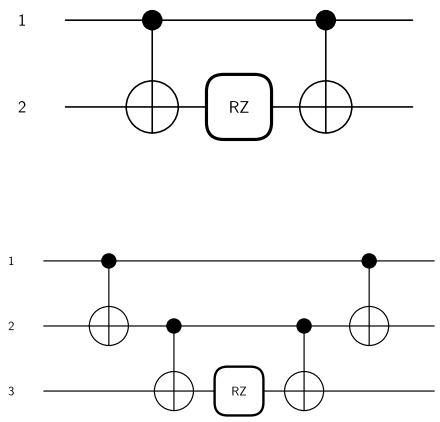

Figure 15 Circuit implementation of the cost unitary for two-qubit and three-qubit terms, i.e. $\exp(i\gamma J_{1,2}\hat{Z}_1\hat{Z}_2)$ and $\exp(i\gamma J_{1,2,3}\hat{Z}_1\hat{Z}_2\hat{Z}_3)$, respectively. The circuit consists of a sequence of CNOT gates to entangle the qubits, such that the initial states $|b_1\rangle_1 |b_2\rangle_2$ and $|b_1\rangle_1 |b_2\rangle_2 |b_3\rangle_3$ in the computational basis become $|b_1\rangle_1 |b_1 \oplus b_2\rangle_2$ and $|b_1\rangle_1 |b_1 \oplus b_2\rangle_2 |b_1 \oplus b_2 \oplus b_3\rangle_3$ through the first chain of CNOT gates. The single qubit rotation $R_Z(2\gamma J_{0,1})$ and $R_Z(2\gamma J_{0,1,2})$ acts on the last qubit. After the rotation, another sequence of CNOT gates disentangles the qubits, resulting in the final state $\exp((-1)^{b_1\oplus b_2} i\gamma J_{1,2}) |b_1\rangle_1 |b_2\rangle_2$ and $\exp((-1)^{b_1\oplus b_2\oplus b_3} i\gamma J_{1,2,3}) |b_1\rangle_1 |b_2\rangle_2 |b_3\rangle_3$

Appendix B: Cost-efficient circuit implementation of HUBO Hamiltonians

An optimized circuit implementation of a HUBO Hamiltonian can be achieved using a Gray code approach, introduced in [76] and [72], which we largely follow. This circuit implementation has been successfully employed to problems like the traveling salesperson problem [26] and the circuit representation of boolean functions [77].

To understand this approach, we examine the effect of the circuit introduced in Sect. 2.3 on a computational basis state $|\mathbf{b}\rangle = \otimes_{t\in T} |b_t\rangle_t$ for a set of qubit indices T. To construct a circuit for the term $\exp(iJ_T\hat{Z}_T)$, we create entangled states where one qubit encodes the parity $\oplus_{t\in T} b_t$. The R_Z gate implements $\exp(-it\hat{Z}/2)$, so acting with an R_Z gate on this qubit results in a phase factor $\exp(i\gamma J_T)$ if $\oplus_{t\in T} b_t = 1$ and $\exp(-i\gamma J_T)$ if $\oplus_{t\in T} b_t = 0$. This is exactly how $\exp(iJ_T\hat{Z}_T)$ acts on a computational basis state. We illustrate the circuit in Fig. 15, where we show the circuit diagrams to implement the terms $\exp(i\gamma J_{1,2}\hat{Z}_1\hat{Z}_2)$ and $\exp(i\gamma J_{1,2,3}\hat{Z}_1\hat{Z}_2\hat{Z}_3)$. Each term $\exp(-i\gamma J_T)$ implemented this way increases the circuit depth by $|T| - 1$. In principle this can be reduced by replacing CNOT ladder circuits with the circuits introduced in [78, 79]. However, this comes at the expense of doubling the CNOT count and introducing ancilla qubits.

To implement all terms of the HUBO-Hamiltonian $\sum_{T\subseteq\{1,\ldots,d\}} J_T\hat{Z}_T$, we construct a circuit that creates parity states $|\oplus_{t\in T} b_t\rangle$ and acts R_Z gates on these states for all subsets $T \subseteq \{1,\ldots,d\}$ at least once. This can, in principle, be done by using circuits like in Fig. 15 sequentially. However, this is suboptimal in the number of CNOT gates. The construction of parity states $|\oplus_{t\in T} b_t\rangle$ for all subsets $T \subseteq \{1,\ldots,d\}$ with minimal CNOT gate costs is a routing problem, and as shown in [76] and [72] the Gray Code implementation is the optimal approach to it.

The key observation for optimizing the circuit is that once the parity state corresponding to a subset $T_1 \subseteq \{1,\ldots,d\}$ is prepared, the parity state for any subset $T_2 \subseteq \{1,\ldots,d\}$ differing from T_1 by exactly one element can be obtained by a single additional operation. In particular, the phase factor $\exp(iJ_{T_2}\hat{Z}_{T_2})$ is implemented by applying a single CNOT gate, with the differing element as control and the qubit encoding $\bigoplus_{a\in T_1} b_a$ as target.

To introduce the Gray Code-based circuit, we denote each subset $T_1 \subseteq \{1,\ldots,d\}$ through a binary string $\mathbf{g}$ of length d, where the j-th element is 1 if $j \in T_1$ and 0 otherwise. If two subsets T_1, T_2 differ in exactly one element, the corresponding binary strings $\mathbf{g}_1, \mathbf{g}_2$ differ in exactly one bit.

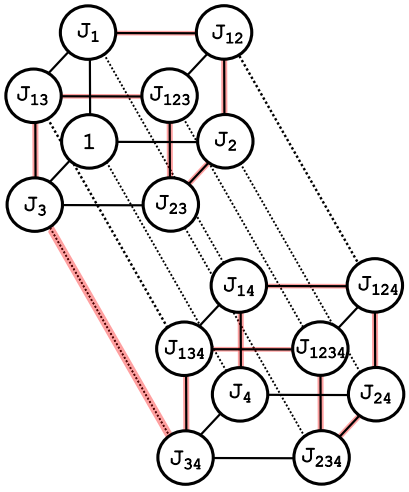

Figure 16 Gray code traversal for optimal implementation of a 4-qubit HUBO Hamiltonian. Each vertex of the 4-dimensional hypercube represents a subset $T \subseteq \{1,2,3,4\}$ of qubits, with subscripts indicating which qubits are included. The red path shows the Gray code sequence starting from the coefficient J_1 (bitstring 1000, the bitstring 0000 corresponds to a global phase and is ignored). Each edge corresponds to flipping exactly one bit, enabling implementation of consecutive terms with only one additional CNOT gate per term. This systematic traversal visits all $2^4 - 1 = 15$ vertices exactly once, requiring only $2^4 - 2 = 14$ CNOT gates total for the complete HUBO Hamiltonian $\sum_{T\subseteq\{1,2,3,4\}} J_T \hat{Z}_T$

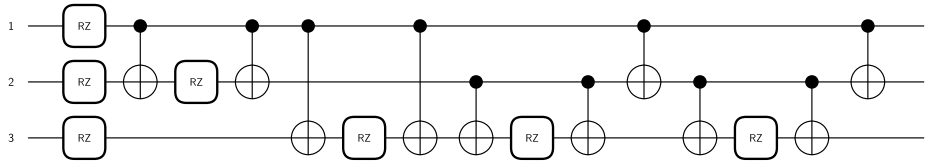

Figure 17 Non-optimized implementation of the cost unitary for a three-qubit HUBO Hamiltonian $\sum_{T\subseteq\{1,2,3\}} J_T \hat{Z}_T$, where T runs over all subsets of the three qubits. Each term is implemented sequentially using parity circuits as shown in Fig. 15, resulting in $2^4(4-2)+2 = 10$ CNOT gates and $2^4 - 1 = 7$ R_Z gates. This method incurs a higher CNOT count than the Gray code-based optimal circuit presented in Fig. 18, highlighting the benefit of circuit optimization for efficient quantum computation

Binary strings of length d map naturally to d-dimensional hypercubes, and implementing all subsets of $\{1,\ldots,d\}$ is equivalent to traversing all vertices of this hypercube. Each vertex of the hypercube corresponds to one binary string and therefore to one subset $T \subseteq \{1,\ldots,d\}$. Two neighboring vertices differ by one element. The construction of parity states $|\oplus_{i\in T} b_i\rangle$ for all subsets $T \subseteq \{1,\ldots,d\}$ with minimal CNOT gate costs, now amounts to traversing the hypercube with as few steps as possible.

The Gray code provides a systematic way to traverse the vertices of this hypercube. Crucially, the Gray code visits each vertex of the hypercube exactly once, thereby avoiding any redundant operations or inefficiencies that would arise from revisiting previously implemented terms. Starting from the vertex corresponding to the global phase, i.e., the bitstring $\mathbf{0} = (0,0,\ldots,0)$ and following the Gray code sequence, we can implement all terms in the HUBO Hamiltonian using only one CNOT gate and one R_Z gate per term. This is illustrated in Fig. 16 for a four-qubit HUBO Hamiltonian. Welch et al. [72] have proven that this Gray code implementation is optimal in the sense that it requires the minimum number of CNOT gates to implement all terms in the HUBO Hamiltonian.

In practice, this means that a HUBO Hamiltonian of the form $\sum_{T\subseteq\{1,\ldots,d\}} J_T \hat{Z}_T$ can be implemented with $2^{|d|} - 2$ CNOT gates and $2^{|d|} - 1$ R_Z gates. The sequential approach requires $2^{|d|}(|d|-2)+2$ CNOT gates. An illustration of the optimized circuit for the three-qubit HUBO Hamiltonian, in comparison to the non-optimized implementation shown in Fig. 17, is presented in Fig. 18.

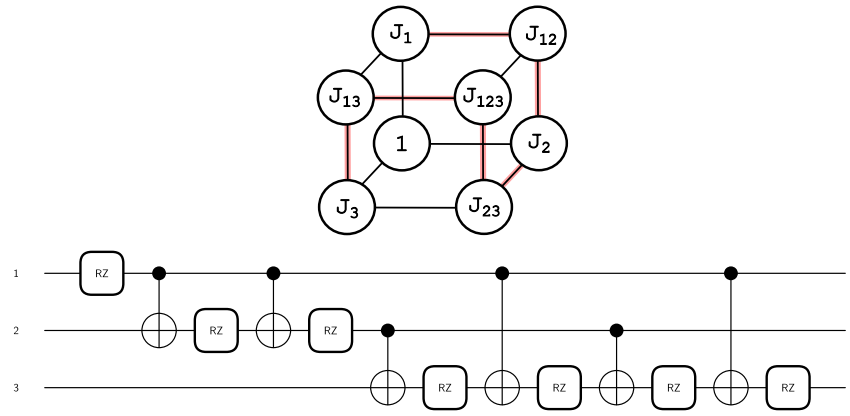

Figure 18 Optimal quantum circuit for implementing the cost unitary of a three-qubit HUBO Hamiltonian using the Gray code approach. Here, each subset $T \subseteq \{1,2,3\}$ defines a term $J_T \hat{Z}_T$ in the Hamiltonian. The upper panel illustrates the Gray code traversal of the three-dimensional hypercube, enabling efficient implementation with one additional CNOT and one R_Z gate per term. For three qubits, this results in $2^3 - 2 = 6$ CNOT gates and $2^3 - 1 = 7$ R_Z gates for all nontrivial subsets T, demonstrating the reduction in CNOT gates achieved by this optimized construction. Compared to the non-optimized implementation shown in Fig. 17, this approach significantly reduces the CNOT gate count

Appendix C: Derivation: scaling of quantum resources for QUBO and HUBO formulations

In this section, we derive scaling relations for the quantum resources required to encode a quadratic COP in the QUBO and HUBO encodings. The COP we consider has a quadratic objective function of the form

$$C(\mathbf{s}) = \sum_{1 \le i \le n} c_1(i, s_i) + \sum_{1 \le i < j \le n} c_2(i, j, s_i, s_j). \tag{31}$$

We will consider the most general instance of this problem, where the linear coefficients $c_1(i,v)$ are nonzero for all variables i and all values v, and the quadratic coefficients $c_2(i,j,v,w)$ are nonzero for all variable combinations $i \neq j$ and all value pairs v, w. This instance then has $n \cdot m$ linear coefficients and $\binom{n}{2} \cdot m^2$ quadratic coefficients. Of course, in practice, the objective function often has a simpler form, but here we consider the worst-case scaling of the quantum resources required for the problem.

C.1 QUBO formulation

The QUBO encoding uses $n \cdot m$ qubits, where each qubit represents a unique variable-value pair. The linear coefficients in the objective function give rise to one-body terms in the cost Hamiltonian. Since each one-body term is implemented using a single R_Z gate, the number of R_Z gates required to implement all linear terms is equal to the number of qubits.

The quadratic coefficients translate to two-body terms in the cost Hamiltonian, each requiring a circuit with two CNOT gates and one R_Z gate (see Fig. 15, left circuit). Since there are $\binom{n}{2} \cdot m^2$ quadratic coefficients, the number of CNOT gates required to implement all quadratic terms is $2 \cdot \binom{n}{2} \cdot m^2$, and the number of R_Z gates is $\binom{n}{2} \cdot m^2$. The asymptotic scaling of the number of CNOT and R_Z gates is therefore $\mathcal{O}(n^2 m^2)$.

C.2 HUBO encoding

In the HUBO encoding, each variable is represented by $d = \lceil \log_2(m) \rceil$ qubits. Therefore, the total number of qubits required is $n \cdot \lceil \log_2(m) \rceil$. The linear coefficients in the objective function give rise to a Hamiltonian term $\sum_{T \subseteq T_i} J_T \hat{Z}_T$, where T_i denotes the set of d qubits encoding variable i. This generates many-body interactions ranging from single-qubit terms (1st order) up to d-qubit terms (dth order) for each of the n variables.

The quadratic coefficients for variable combinations $i \neq j$ produce a Hamiltonian term $\sum_{T_1 \subseteq T_i, T_2 \subseteq T_j} J_{T_1,T_2} \hat{Z}_{T_1 \cup T_2}$. Here, T_1 and T_2 are subsets of the qubit indices, T_i and T_j, encoding variables i and j, respectively. Importantly, since $T_i \subseteq T_i \cup T_j$, the linear terms can be encoded simultaneously with the quadratic terms, eliminating the need for separate linear term implementation. As discussed in Appendix B, implementing the coefficients for all subsets of a set of qubit indices T requires $2^{|T|} - 1$ R_Z gates and $2^{|T|} - 2$ CNOT gates. Since the union $T_i \cup T_j$ involves $2d$ qubits, this translates to $2^{2d} - 1$ R_Z gates and $2^{2d} - 2$ CNOT gates per variable pair. With $\binom{n}{2}$ variable pairs, the total number of R_Z gates required is $\binom{n}{2} \cdot (2^{2d} - 1)$ and the total number of CNOT gates is $\binom{n}{2} \cdot (2^{2d} - 2)$. With $d = \lceil \log_2(m) \rceil$, this gives us the asymptotic scaling of the CNOT and R_Z gates as $\mathcal{O}(n^2 m^2)$. In contrast, the sequential implementation requires $2^{2d}(2d-1) + 2$ CNOT gates per variable pair. The CNOT gate count in the sequential implementation thus scales as $\mathcal{O}(n^2 m^2 \log(m))$.

Appendix D: Technical details on QAOA experiments

The exact computation of the approximation ratio in Eq. (25) requires summing over all feasible solutions, a process that scales exponentially with problem size in the considered benchmarks. Therefore, we estimate the approximation ratio using sampling-based methods. Specifically, the sum over all feasible states is replaced by a sum over sampled bitstrings,

$$\sum_{\mathbf{b} \in B_{\text{feas}}} \rightarrow \sum_{\mathbf{b} \in B_{\text{sample}}} \delta(\mathbf{b}), \tag{32}$$

where $\delta(\mathbf{b})$ equals one if the bitstring $\mathbf{b}$ satisfies all constraints, and zero otherwise. The required sample complexity is determined using Hoeffding's inequality [80]. Employing $|B_{\text{sample}}| = 10{,}000$ samples in all experiments ensures a 99% probability that the estimated value is within 2% of the true approximation ratio.

Furthermore, for all benchmarked COP instances and QAOA depths, we performed 100 independent QAOA runs. This ensures that all reported averages and standard deviations are directly comparable across experiments. For the target approximation benchmarks, we ran the QAOA algorithm 100 times and picked the QAOA experiment with the lowest layer requirements to reach the target threshold.

D.1 Percentage reduction calculations

Throughout this work, percentage reductions in gate counts are computed as:

$$\text{Reduction } \% = 100 \times \frac{\text{Gate count}_{\text{QUBO}} - \text{Gate count}_{\text{HUBO}}}{\text{Gate count}_{\text{QUBO}}}, \tag{33}$$

Table 3 Per-layer gate counts (CNOT, R_Z, Hadamard, R_X) for all benchmarked problem instances and sizes, comparing QUBO and HUBO encodings

Problem Instance	QUBO				HUBO			
	CNOT	R_Z	Hadamard	R_X	CNOT	R_Z	Hadamard	R_X
Gate Assignment Problem (GAP, 4 gates fixed)								
GAP (1 flight)	12	10	4	4	0	1	2	2
GAP (2 flights)	32	24	8	8	10	5	4	4
GAP (3 flights)	76	50	12	12	34	14	6	6
GAP (4 flights)	96	64	16	16	44	18	8	8
GAP (5 flights)	140	90	20	20	68	27	10	10
Maximum k-Colorable Subgraph (MkCS, 4 colors fixed)								
MkCS (1 vertex)	12	10	4	4				
MkCS (2 vertices)	32	24	8	8	10	3	4	4
MkCS (3 vertices)	52	38	12	12	20	6	6	6
MkCS (4 vertices)	88	60	16	16	50	15	8	8
MkCS (5 vertices)	132	86	20	20	90	27	10	10
Integer Programming (IP, 4 values fixed)								
IP (1 variable)	12	10	4	4	0	2	2	2
IP (2 variables)	42	29	8	8	8	7	4	4
IP (3 variables)	90	57	12	12	24	18	6	6
IP (4 variables)	120	76	16	16	38	31	8	8
IP (5 variables)	150	95	20	20	46	37	10	10

where a reduction of 100% occurs when $\text{Gate count}_{\text{HUBO}} = 0$, as happens for single-variable instances when no CNOT gates are required. All reported percentage reductions refer to per-layer gate counts. Detailed per-layer gate counts (CNOT, R_Z, Hadamard, and R_X) for all benchmarked problem sizes and both encodings are provided in Table 3. The Gray code circuit implementation is more efficient than the sequential implementation for a full HUBO Hamiltonian $\sum_{T\subseteq\{1,\dots,d\}} J_T \hat{Z}_T$. However, when there are zero HUBO coefficients J_T for some sets T then in principle the sequential implementation can also be more CNOT. For the MkCS instance with one vertex, we omit the HUBO scaling because, in the HUBO encoding, any sampled two-bit string is already optimal. Consequently, this instance can be solved with a zero-layer QAOA circuit.

D.2 One-hot constraint contribution in QUBO encoding

To understand the source of gate-count reduction, we report in Table 4 the per-layer CNOT and R_Z gate counts required solely for implementing the one-hot penalty terms in the QUBO encoding. The one-hot penalty $\lambda(1-\sum_{v=1}^{m} x_{i,v})^2$ for each variable expands to m linear terms and $m(m-1)$ two-body terms. For all three problems with n variables and values $m = 4$, this yields $12n$ CNOT gates and $10n$ RZ gates per variable, independent of the problem type. Note that in the full QUBO encoding, some of these gates may be shared or merged with gates implementing the objective function and problem-specific constraints; therefore, the total gate count is not simply the sum of the one-hot and problem-specific contributions.

Appendix E: Details on the GAP instance

For benchmarking, we use a GAP instance with $n = 5$ flights and $m = 4$ airport-gates, numbered 1 to 4, as shown in Fig. 6. The numbers of arriving and departing passengers per flight are given in Table 6, with values ranging from 61 to 88. Out of the total, there are 13 transfer passengers between flights 1 and 4, and 29 transfer passengers between flights

Table 4 Per-layer CNOT and R_Z gate counts for one-hot penalty terms in the QUBO encoding. These counts are identical for all three benchmarked problems (GAP, MkCS, and IP) since they all use $m = 4$ values per variable

Number of Variables	CNOT	R_Z
1	12	10
2	24	20
3	36	30
4	48	40
5	60	50

Table 5 Walking times (in minutes) between check-in/luggage claim and airport gates 1-4 for the benchmarked GAP instance

	Check-in/Luggage claim	Gate 1	Gate 2	Gate 3	Gate 4
Check-in/Luggage claim	0	10	10	20	20
Gate 1	10	0	20	20	20
Gate 2	10	20	0	20	20
Gate 3	20	20	20	0	1
Gate 4	20	20	20	1	0

Table 6 Number of arriving and departing passengers for each of the five aircraft in the benchmarked GAP instance

Aircraft i	$p_i^{arr} + p_i^{dep}$
0	75
1	74
2	62
3	88
4	61

0 and 2. Transfer passengers contribute to the quadratic cost term $c_2(i, j, v, w)$ in Eq. (6), while the remaining passengers are included in the linear term $c_1(i, v)$ in Eq. (5).

The walking times between check-in/luggage claim and the airport-gates, as well as in between the airport-gates, are given in Table 5. In this instance, the walking times from each gate to check-in and from each gate to luggage claim are identical. For this reason, there is no difference in the cost contribution from arriving or departing passengers. Both groups affect the objective function in the same way, hence we don't distinguish between them in Table 6.

Gate assignments are further constrained by overlapping flight schedules. Any flights that overlap cannot be assigned to the same gate. The overlapping flight pairs in this instance are $\{(0,1),\ (1,2),\ (2,3),\ (3,4)\}$.

Acknowledgements
For this work, the HPC-cluster Hummel-2 at University of Hamburg was used. The cluster was funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 498394658. Für diese Arbeit wurde das HPC-Cluster Hummel-2 an der Universität Hamburg genutzt, das durch die Deutsche Forschungsgemeinschaft (DFG) – 498394658 – gefördert wurde. The authors also thank Felix Herbort for his technical support and Robert Glöckner and Shaham Jafarpisheh for helpful discussions. The authors also thank Igor Gaidai for a thorough reading of the manuscript and valuable feedback. We acknowledge financial support from the Open Access Publication Fund of Universität Hamburg.

Author contributions
FK developed the required codes for the calculations and plotted the graphs. FK and SP analyzed the data and wrote the manuscript. RM, JD, and DJ provided comments for improving the manuscript. JD and DJ are the supervisors of the project.

Funding information
Open Access funding enabled and organized by Projekt DEAL. The authors gratefully acknowledge the funding provided by the Hamburgische Investitions- und Förderbank for the project "Efficient Quantum Algorithms for Aviation". SP and DJ

acknowledge support from the Hamburg Quantum Computing Initiative (HQIC) project EFRE. The EFRE project is co-financed by ERDF of the European Union and by "Fonds of the Hamburg Ministry of Science, Research, Equalities and Districts (BWFGB)". DJ acknowledges support from the Cluster of Excellence 'Advanced Imaging of Matter' of the Deutsche Forschungsgemeinschaft (DFG) - EXC 2056 - project ID 390715994, the European Union's Horizon Programme (HORIZON-CL42021-DIGITALEMERGING-02-10) Grant Agreement 101080085 QCFD, and DFG project 'Quantencomputing mit neutralen Atomen'(JA 1793/1-1, Japan-JST-DFG-ASPIRE 2024). DJ acknowledges funding from the Federal Ministry of Research, Technology, and Space (BMFTR) under the grant BeRyQC.

Data availability
The datasets generated and/or analysed during the current study are available in the UHH Forschungsdatenrepositorium repository, https://doi.org/10.25592/uhhfdm.18252.

Declarations

Competing interests
The authors declare no competing interests.

Author details
[1]Institute for Quantum Physics, University of Hamburg, Luruper Chaussee 149, 22761, Hamburg, Germany. [2]Max Planck Institute for the Structure and Dynamics of Matter, Luruper Chaussee 149, 22761, Hamburg, Germany. [3]Zentrum für Optische Quantentechnologien, University of Hamburg, Luruper Chaussee 149, 22761, Hamburg, Germany. [4]Department of Physics & Astronomy, University of Tennessee, Chattanooga, 37403, TN, USA. [5]UTC Quantum Center, University of Tennessee, Chattanooga, 37403, TN, USA. [6]Lufthansa Industry Solutions, Südportal 7, Norderstedt, 22848, Schleswig-Holstein, Germany. [7]The Hamburg Centre for Ultrafast Imaging, University of Hamburg, Luruper Chaussee 149, 22761, Hamburg, Germany. [8]Clarendon Laboratory, University of Oxford, Parks Road, OX1 3PU, Oxford, UK.

Received: 23 January 2026 Accepted: 18 May 2026 Published online: 25 May 2026

References

1. Marzec M. Portfolio optimization: applications in quantum computing. New York: Wiley; 2016. Available from https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118593486.ch4.
2. Perdomo-Ortiz A, Dickson N, Drew-Brook M, Rose G, Aspuru-Guzik A. Finding low-energy conformations of lattice protein models by quantum annealing. Sci Rep. 2012;2(1):571. https://doi.org/10.1038/srep00571.
3. Boucherie RJ, Braaksma A, Tijms H. Operations research. World Scientific; 2021.
4. Crescenzi P, Kann V. A compendium of NP optimization problems. Berlin: Springer; 1995. Available from https://cs.pwr.edu.pl/zielinski/lectures/om/compendium.pdf.
5. Fu Y, Anderson PW. Application of statistical mechanics to NP-complete problems in combinatorial optimisation. J Phys A, Math Gen. 1986;19(9):1605. https://doi.org/10.1088/0305-4470/19/9/033.
6. Kirkpatrick S, Gelatt JCD, Vecchi MP. Optimization by simulated annealing. Science. 1983;220(4598):671–80. https://doi.org/10.1126/science.220.4598.671.
7. Glover F, TaiUard E, de Werra D. A user's guide to tabu search. Ann Oper Res. 1993;41(1):1–28. https://doi.org/10.1007/BF02078647.
8. IBM. IBM ILOG CPLEX Optimization Studio. 2024. Webpage. Available from https://www.ibm.com/products/ilog-cplex-optimization-studio.
9. Gurobi. Gurobi Optimization. 2025. Webpage. Available from https://www.gurobi.com/.
10. Xu L, Liberti L. Relaxations for binary polynomial optimization via signed certificates. 2024. arXiv preprint. Available from https://arxiv.org/abs/2405.13447.
11. Puchinger J, Raidl GR, Pferschy U. The multidimensional knapsack problem: structure and algorithms. INFORMS J Comput. 2010;22(2):250–65. https://doi.org/10.1287/ijoc.1090.0344.
12. Packebusch T, Mertens S. Low autocorrelation binary sequences. J Phys A, Math Theor. 2016;49(16):165001. https://doi.org/10.1088/1751-8113/49/16/165001. arXiv:1512.02475 [cond-mat].
13. Danilova M, Dvurechensky P, Gasnikov A, Gorbunov E, Guminov S, Kamzolov D, et al. Recent theoretical advances in non-convex optimization. In: Nikeghbali A, Pardalos PM, Raigorodskii AM, Rassias MT, editors. High-dimensional optimization and probability: with a view towards data science. Cham: Springer; 2022. p. 79–163.
14. Burer S, Letchford AN. Non-convex mixed-integer nonlinear programming: a survey. Surv Oper Res Manag Sci. 2012;17(2):97–106. https://doi.org/10.1016/j.sorms.2012.08.001.
15. Floudas CA, Akrotiriankis IG, Caratzoulas S, Meyer CA, Kallrath J. Global optimization in the 21st century: advances and challenges. Comput Chem Eng. 2004;29(6):1185–202. https://doi.org/10.1016/j.compchemeng.2005.02.006.
16. Albash T, Lidar DA. Adiabatic quantum computation. Rev Mod Phys. 2018;90(1):015002. https://doi.org/10.1103/RevModPhys.90.015002. arXiv:1611.04471 [quant-ph].
17. Ebadi S, Keesling A, Cain M, Wang TT, Levine H, Bluvstein D, et al. Quantum optimization of maximum independent set using Rydberg atom arrays. Science. 2022;376(6598):1209–15. https://doi.org/10.1126/science.abo6587. arXiv:2202.09372 [quant-ph].
18. Farhi E, Goldstone J, Gutmann S. A quantum approximate optimization algorithm. 2014. arXiv preprint. Available from https://arxiv.org/abs/1411.4028.
19. Lucas A. Ising Formulations of Many NP Problems. Front Phys. 2014;2. https://doi.org/10.3389/fphy.2014.00005.
20. Goswami K, Mukherjee R, Ott H, Schmelcher P. Solving optimization problems with local light shift encoding on Rydberg quantum annealers. Phys Rev Res. 2024;6(2):023031. https://doi.org/10.1103/PhysRevResearch.6.023031 arXiv:2308.07798 [quant-ph].
21. Lai CT, Blank C, Schmelcher P, Mukherjee R. Towards arbitrary QUBO optimization: analysis of classical and quantum-activated feedforward neural networks. Mach Learn: Sci Technol. 2025;6(2):025049. https://doi.org/10.1088/2632-2153/addb97.

22. Zaman M, Tanahashi K, Tanaka S. PyQUBO: Python library for mapping combinatorial optimization problems to QUBO form. IEEE Trans Comput. 2022;71(4):838–50. https://doi.org/10.1109/TC.2021.3063618.
23. Dominguez F, Unger J, Traube M, Mant B, Ertler C, Lechner W. Encoding-independent optimization problem formulation for quantum computing. Front Quantum Sci Technol. 2023. https://doi.org/10.3389/frqst.2023.1229471.
24. Montañez-Barrera JA, Willsch D, Maldonado-Romo A, Michielsen K. Unbalanced penalization: a new approach to encode inequality constraints of combinatorial problems for quantum optimization algorithms. Quantum Sci Technol. 2024;9(2):025022. https://doi.org/10.1088/2058-9565/ad35e4.
25. Romero SV, Visuri AM, Gomez Cadavid A, Simen A, Solano E, Hegade NN. Bias-field digitized counterdiabatic quantum algorithm for higher-order binary optimization. Commun Phys. 2025;8(1). https://doi.org/10.1038/s42005-025-02270-3.
26. Glos A, Krawiec A, Zimborás Z. Space-efficient binary optimization for variational quantum computing. npj Quantum Inf. 2022;8(1):39. https://doi.org/10.1038/s41534-022-00546-y.
27. Romero SV, Gomez Cadavid A, Nikačević P, Solano E, Hegade NN, Lopez-Ruiz MA, et al. Protein folding with an all-to-all trapped-ion quantum computer. 2025. arXiv Preprint. Available from https://arxiv.org/abs/2506.07866.
28. Yahui C, Epifanovsky E, Jansen K, Kaushik A, Kühn S. Simulating the flight gate assignment problem on a trapped ion quantum computer. 2023. arXiv preprint. Available from https://arxiv.org/abs/2309.09686.
29. Nagies S, Geier KT, Akram J, Bantounas D, Johanning M, Hauke P. Boosting quantum annealing performance through direct polynomial unconstrained binary optimization. Quantum Sci Technol. 2025;10(3):035008. https://doi.org/10.1088/2058-9565/adcae6. arXiv:2412.04398 [quant-ph].
30. Wintersperger K, Dommert F, Ehmer T, Hoursanov A, Klepsch J, Mauerer W, et al. Neutral atom quantum computing hardware: performance and end-user perspective. EPJ Quantum Technol. 2023;10(1). https://doi.org/10.1140/epjqt/s40507-023-00190-1.
31. Fauseweh B. Quantum many-body simulations on digital quantum computers: state-of-the-art and future challenges. Nat Commun. 2024;15(1):2123. https://doi.org/10.1038/s41467-024-46402-9.
32. Koch F. PyHUBO; 2025. Github Repository. Available from https://github.com/frederikKoch/PyHUBO.
33. Schrijver A. Combinatorial Optimization: Polyhedra and Efficiency. Vol. B. J Comput Syst Sci. 2003. Available from https://link.springer.com/book/9783540443896.
34. Crainic TG, Gendreau M, Frangioni A, editors. Combinatorial optimization and applications: a tribute to Bernard gendron. International series in operations research & management science. vol. 358. Cham Springer; 2024.
35. Salehi O, Glos A, Miszczak JA. Unconstrained binary models of the travelling salesman problem variants for quantum optimization. Quantum Inf Process. 2022;21(2):67. https://doi.org/10.1007/s11128-021-03405-5.
36. López-Ibáñez M, Blum C, Ohlmann JW, Thomas BW. The travelling salesman problem with time windows: adapting algorithms from travel-time to makespan optimization. Appl Soft Comput. 2013;13(9):3806–15. https://doi.org/10.1016/j.asoc.2013.05.009.
37. Chen J, Westerheim H, Holmes Z, Luo I, Nuradha T, Patel D, et al. Slack-variable approach for variational quantum semidefinite programming. Phys Rev A. 2025;112(2):022607. https://doi.org/10.1103/lwxq-4myj.
38. Glover F, Kochenberger G, Du Y. Quantum bridge analytics I: a tutorial on formulating and using QUBO models. Ann Oper Res. 2022;314(1):141–83. https://doi.org/10.1007/s10479-022-04634-2.
39. Bouras A, Ghaleb MA, Suryahatmaja US, Salem AM. The airport gate assignment problem: a survey. Sci World J. 2014;2014:1–27. https://doi.org/10.1155/2014/923859.
40. Chai Y, Funcke L, Hartung T, Jansen K, Kühn S, Stornati P, et al. Towards finding an optimal flight gate assignment on a digital quantum computer. Phys Rev Appl. 2023;20(6):064025. https://doi.org/10.1103/PhysRevApplied.20.064025 arXiv:2302.11595 [quant-ph].
41. Bentert M, van Bevern R, Niedermeier R. Inductive *k*-independent graphs and *c*-colorable subgraphs in scheduling: a review. J Sched. 2019;22(1):3–20. https://doi.org/10.1007/s10951-018-0595-8. arXiv:1712.06481 [cs].
42. Halldórsson MM, Halpern JY, Li LE, Mirrokni VS. On spectrum sharing games. In: Proceedings of the twenty-third annual ACM symposium on principles of distributed computing. St. John's Newfoundland Canada: ACM; 2004. p. 107–14.
43. Hertz A, Montagné R, Gagnon F. Constructive algorithms for the partial directed weighted improper coloring problem. J Graph Algorithms Appl. 2016;20(2):159–88. https://doi.org/10.7155/jgaa.00389.
44. Koster AMCA, Scheffel M. A routing and network dimensioning strategy to reduce wavelength continuity conflicts in all-optical networks. vol. 10032. Optimization Online; 2006. Available from https://optimization-online.org/?p=10032.
45. Liu D, Li J, Cheng X, Zhang S, Chang Y, Yan L. Efficient hybrid variational quantum algorithm for solving graph coloring problem. 2025. arXiv preprint. Available from https://arxiv.org/abs/2504.21335.
46. Quintero R, Bernal D, Terlaky T, Zuluaga LF. Characterization of QUBO reformulations for the maximum *k*-colorable subgraph problem. 2021. arXiv preprint. Available from https://arxiv.org/abs/2101.09462.
47. Wang Z, Rubin NC, Dominy JM, Rieffel EG. *XY*-mixers: analytical and numerical results for QAOA. Phys Rev A. 2020;101(1):012320. https://doi.org/10.1103/PhysRevA.101.012320 arXiv:1904.09314 [quant-ph].
48. Streif M, Leib M, Wudarski F, Rieffel E, Wang Z. Quantum algorithms with local particle number conservation: noise effects and error correction. Phys Rev A. 2021;103(4):042412. https://doi.org/10.1103/PhysRevA.103.042412. arXiv:2011.06873 [quant-ph].
49. Sotirov R, Kuryatnikova O, Vera J. The maximum *k*-colorable subgraph problem and related problems. 2021. arXiv preprint. Available from https://arxiv.org/abs/2001.09644.
50. Wolsey L. Integer programming. In: Integer programming. New York: Wiley; 2020. p. 1–23.
51. Yves P, Laurence AW. Production planning by mixed integer programming. Springer series in operations research and financial engineering. New York: Springer; 2006.
52. Magatão L, Arruda LVR, Neves Jr F. A mixed integer programming approach for scheduling commodities in a pipeline. In: Grievink J, van Schijndel J, editors. European symposium on computer aided process engineering-12. Computer aided chemical engineering. vol. 10. Amsterdam: Elsevier; 2002. p. 715–20.
53. Goswami K, Schmelcher P, Mukherjee R. Qudit-based scalable quantum algorithm for solving the integer programming problem. 2025. arXiv preprint. Available from https://arxiv.org/abs/2508.13906.

54. Svensson M, Andersson M, Grönkvist M, Vikstål P, Dubhashi D, Ferrini G, et al. Hybrid quantum-classical heuristic to solve large-scale integer linear programs. Phys Rev Appl. 2023;20(3):034062. https://doi.org/10.1103/PhysRevApplied.20.034062.
55. Sharma M, Lau HC. Cutting Slack: quantum optimization with slack-free methods for combinatorial benchmarks. 2025. arXiv preprint. Available from https://arxiv.org/abs/2507.12159.
56. Tanahashi K, Takayanagi S, Motohashi T, Tanaka S. Application of Ising machines and a software development for Ising machines. J Phys Soc Jpn. 2019;88(6):061010. https://doi.org/10.7566/JPSJ.88.061010.
57. Hadfield S. On the representation of Boolean and real functions as Hamiltonians for quantum computing. ACM Trans Quantum Comput. 2021;2(4):1–21. https://doi.org/10.1145/3478519. arXiv:1804.09130 [quant-ph].
58. Farhi E, Goldstone J, Gutmann S, Lapan J, Lundgren A, Preda D. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. Science. 2001;292(5516):472–5. https://doi.org/10.1126/science.1057726. arXiv:quant-ph/0104129.
59. McArdle S, Jones T, Endo S, Li Y, Benjamin SC, Yuan X. Variational Ansatz-based quantum simulation of imaginary time evolution. npj Quantum Inf. 2019;5(1):75. https://doi.org/10.1038/s41534-019-0187-2.
60. Beach MJS, Melko RG, Grover T, Hsieh TH. Making trotters sprint: a variational imaginary time ansatz for quantum many-body systems. Phys Rev B. 2019;100(9):094434. https://doi.org/10.1103/PhysRevB.100.094434.
61. Love PJ. Cooling with imaginary time. Nat Phys. 2020;16(2):130–1. https://doi.org/10.1038/s41567-019-0709-z.
62. Peruzzo A, McClean J, Shadbolt P, Yung MH, Zhou XQ, Love PJ, et al. A variational eigenvalue solver on a photonic quantum processor. Nat Commun. 2014;5(1):4213. https://doi.org/10.1038/ncomms5213.
63. Tilly J, Chen H, Cao S, Picozzi D, Setia K, Li Y, et al. The variational quantum eigensolver: a review of methods and best practices. Phys Rep. 2022;986:1–128. https://doi.org/10.1016/j.physrep.2022.08.003 arXiv:2111.05176 [quant-ph].
64. Zhou L, Wang ST, Choi S, Pichler H, Lukin MD. Quantum approximate optimization algorithm: performance, mechanism, and implementation on near-term devices. Phys Rev X. 2020;10(2):021067. https://doi.org/10.1103/PhysRevX.10.021067.
65. Basso J, Gamarnik D, Mei S, Zhou L. Performance and limitations of the QAOA at constant levels on large sparse hypergraphs and spin glass models. In: 2022 IEEE 63rd annual symposium on foundations of computer science (FOCS). 2022. p. 335–43.
66. Blekos K, Brand D, Ceschini A, Chou CH, Li RH, Pandya K, et al. A review on quantum approximate optimization algorithm and its variants. Phys Rep. 2024;1068:1–66. https://doi.org/10.1016/j.physrep.2024.03.002. arXiv:2306.09198 [quant-ph].
67. Golden J, Bärtschi A, O'Malley D, Eidenbenz S. Numerical evidence for exponential speed-up of QAOA over unstructured search for approximate constrained optimization. In: 2023 IEEE international conference on quantum computing and engineering (QCE). 2023. p. 496–505.
68. Weidenfeller J, Valor LC, Gacon J, Tornow C, Bello L, Woerner S, et al. Scaling of the quantum approximate optimization algorithm on superconducting qubit based hardware. Quantum. 2022;6:870. https://doi.org/10.22331/q-2022-12-07-870. arXiv:2202.03459 [quant-ph].
69. Kurowski K, Pecyna T, Slysz M, Różycki R, Waligóra G, Węglarz J. Application of quantum approximate optimization algorithm to job shop scheduling problem. Eur J Oper Res. 2023;310(2):518–28. https://doi.org/10.1016/j.ejor.2023.03.013.
70. Wang SS, Liu HL, Song YQ, Gao F, Qin SJ, Wen QY. Quantum alternating operator ansatz for solving the minimum exact cover problem. Phys A, Stat Mech Appl. 2023;626:129089. https://doi.org/10.1016/j.physa.2023.129089.
71. Sachdeva N, Hartnett GS, Maity S, Marsh S, Wang Y, Winick A, et al. Quantum optimization using a 127-qubit gate-model IBM quantum computer can outperform quantum annealers for nontrivial binary optimization problems. 2024. arXiv preprint. Available from https://arxiv.org/abs/2406.01743.
72. Welch J, Greenbaum D, Mostame S, Aspuru-Guzik A. Efficient quantum circuits for diagonal unitaries without ancillas. New J Phys. 2014;16(3):033040. https://doi.org/10.1088/1367-2630/16/3/033040.
73. Blondel M, Berthet Q, Cuturi M, Frostig R, Hoyer S, Llinares-López F, et al. Efficient and modular implicit differentiation. 2022. arXiv Preprint. Available from https://arxiv.org/abs/2105.15183.
74. Bergholm V, Izaac J, Schuld M, Gogolin C, Ahmed S, Ajith V, et al. PennyLane: automatic differentiation of hybrid quantum-classical computations. 2022. arXiv preprint. Available from https://arxiv.org/abs/1811.04968.
75. Schulz S, Willsch D, Michielsen K. Guided quantum walk. Phys Rev Res. 2024;6(1):013312. https://doi.org/10.1103/PhysRevResearch.6.013312. arXiv:2308.05418 [quant-ph].
76. Verchère Z, Elloumi S, Simonetto A. Optimizing Variational Circuits for Higher-Order Binary Optimization. 2023. arXiv preprint. Available from https://arxiv.org/abs/2307.16756.
77. Amy M, Azimzadeh P, Mosca M. On the CNOT-complexity of CNOT-PHASE circuits. Quantum Sci Technol. 2018;4(1):015002. https://doi.org/10.1088/2058-9565/aad8ca. arXiv:1712.01859 [quant-ph].
78. Bäumer E, Woerner S. Measurement-based long-range entangling gates in constant depth. Phys Rev Res. 2025;7(2):023120. https://doi.org/10.1103/PhysRevResearch.7.023120.
79. Tserkis S, Umer M, Angelakis DG. Depth optimization of CNOT ladder circuits; 2026. Available from arXiv:2511.13256 [quant-ph].
80. Hoeffding W. Probability inequalities for sums of bounded random variables. J Am Stat Assoc. 1963;58(301):13–30. https://doi.org/10.1080/01621459.1963.10500830.

Publisher's note