# Polynomial Reduction Methods and their Impact on QAOA Circuits

Lukas Schmidbauer
*Technical University of Applied Sciences Regensburg*
Regensburg, Germany
lukas.schmidbauer@othr.de

Karen Wintersperger
*Siemens AG, Technology*
Munich, Germany
karen.wintersperger@siemens.com

Elisabeth Lobe
*German Aerospace Center (DLR), Institute of Software Technology, Department High-Performance Computing*
Braunschweig, Germany
elisabeth.lobe@dlr.de

Wolfgang Mauerer
*Technical University of Applied Sciences Regensburg*
*Siemens AG, Technology*
Regensburg/Munich, Germany
wolfgang.mauerer@othr.de

***Abstract*—Abstraction layers are of paramount importance in software architecture, as they shield the higher-level formulation of payload computations from lower-level details. Since quantum computing (QC) introduces many such details that are often unaccustomed to computer scientists, an obvious desideratum is to devise appropriate abstraction layers for QC. For discrete optimisation, one such abstraction is to cast problems in quadratic unconstrained binary optimisation (QUBO) form, which is amenable to a variety of quantum approaches. However, different mathematically equivalent forms can lead to different behaviour on quantum hardware, ranging from ease of mapping onto qubits to performance scalability.**

**In this work, we show how using higher-order problem formulations (that provide better expressivity in modelling optimisation tasks than plain QUBO formulations) and their automatic transformation into QUBO form can be used to leverage such differences to prioritise between different desired non-functional properties for quantum optimisation. Based on a practically relevant use-case and a graph-theoretic analysis, we evaluate how different transformation approaches influence widely used quantum performance metrics (circuit depth, gates count, gate distribution, qubit scaling), and also consider the classical computational efforts required to perform the transformations, as they influence possibilities for achieving future quantum advantage. Furthermore, we establish more general properties and invariants of the transformation methods. Our quantitative study shows that the approach allows us to satisfy different trade-offs, and suggests various possibilities for the future construction of general-purpose abstractions and automatic generation of useful quantum circuits from high-level problem descriptions.**

***Index Terms*—Quantum Software, QAOA, Graphs, Pseudo Boolean Function, QUBO, PUBO**

## I. Introduction

Optimisation problems occur in the realm of practically relevant problems, such as finding optimal time-schedules, production planning, quality control, portfolio optimisation, social network analysis, protein folding and drug discovery. Quantum computers promise complexity theoretical advantages in this field, from quadratic acceleration for finding optimal elements in unstructured search spaces [1] via possible speedups using adiabatic quantum computing and annealing [2], to in-principal super-polynomial speedups for approximate combinatorial optimisation established using advanced reasoning [3].

Formulating an Optimisation Problem (OP) is an essential prerequisite to enjoying possible advantages of large classes of quantum algorithms. On the one hand, one could argue that such formulations are purely mathematical, and are therefore a welcome abstraction layer that decouples the use of quantum algorithms from knowledge of the inner working of quantum systems. On the other hand, the performance of quantum algorithms usually strongly depends on the concrete formulation [4]–[7]. In contrast to noiseless classical computers, this dependence is partly a consequence of current Noisy Intermediate Scale Quantum (NISQ) devices, which suffer from errors induced during execution—invalidating the quantum state [8], [9]. Therefore, the question arises to what extent the degrees of freedom in the formulation of optimisation problems impact the performance of the quantum algorithms and thus to what extent these processes can be decoupled in the overlying design automation process. Fig. 1 illustrates the higher-level process of formulating a real world problem and transforming it to a suitable representation for quantum computers. Here, dashed arrows abstract concrete procedures or algorithms, such as the Quantum Approximate Optimisation Algorithm (QAOA). It also illustrates the present, yet undesirable, coupling of domains through non-functional requirements.

QAOA is a widely used approach to solve combinatorial OPs [10] on gate-based quantum computers. To do so, the OPs are transformed into Quadratic Unconstrained Binary Optimisation (QUBO) problems, or equivalently, except for an affine transformation, Ising models [11], [12] (apart from QAOA, it is well known that other approaches like Variational Quantum Eigensolver (VQE) [13] or quantum annealing (inspired) approaches [4], [14], [15] can solve QUBO problems). The goal of a QUBO problem is to find $\vec{x}^* \in \{0,1\}^n$ which solves

$$\min f(\vec{x}) \text{ such that } \vec{x} \in \{0,1\}^n \tag{1}$$

for a given Pseudo-Boolean Function (PBF) $f : \{0,1\}^n \to \mathbb{R}$, where $f$ has a degree of at most 2. The generalisation for functions $f$ of arbitrary degree is called Polynomial Unconstrained Binary Optimisation (PUBO).

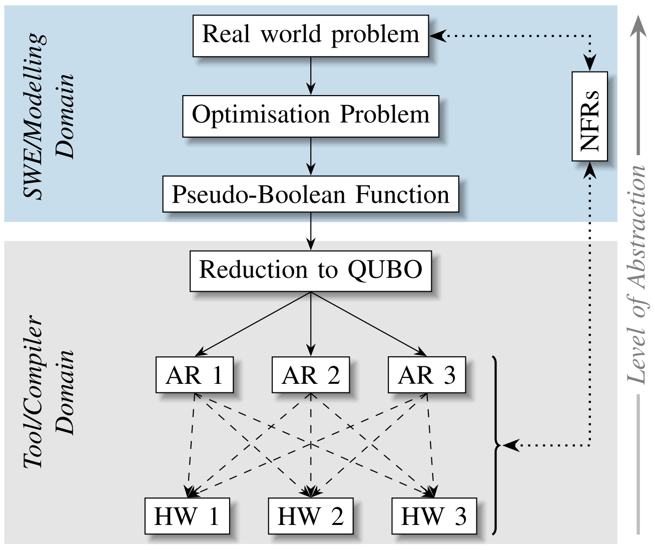

Figure 1: Influence of reduction process and quantum hardware properties on non-functional requirements (NFRs) across domains. AR denotes algebraic representations of a QUBO problem, HW represents transpilation results to different quantum backends. Dashed line: transpilation process, dotted line: dependency.

For practically relevant problem sizes, it is hard to stay within the restrictions imposed by noise and qubit count of NISQ-devices. Nonetheless, it is important to extrapolate findings beyond current HW capabilities to assess their scaling behaviour. Thus, we use metrics, namely the circuit depth, the number of introduced gates, the gate distribution, the number of qubits and the runtime of reductions to analyse the performance of quantum algorithms. Furthermore, current hardware natively supports one- and two-qubit gates. Higher-order gates can be executed via decompositions into a multitude of natively supported gates. Starting from an arbitrary PBF $f$ (*i.e.*, a PUBO) that encodes an OP analogously to above, we investigate two variants to reduce it to a QUBO and arrive at a QAOA circuit, as illustrated by the flow diagram in Fig. 2 (which can also be seen as a more detailed view on the abstraction layer in Fig. 1).

We focus on the problem Hamiltonian $H_P$ of a single layer, since $H_P$ depends on the encoded OP and is therefore the decisive part of our analysis (we deliberately ignore $H_M$, as it would only increase the circuit depth by one). The otherwise $p$-layer deep QAOA circuit [10], [16] produces the state

$$\left|\vec{\beta},\vec{\gamma}\right\rangle = e^{-i\beta_p H_M} e^{-i\gamma_p H_P} \cdots\ e^{-i\beta_1 H_M} e^{-i\gamma_1 H_P} \left|s\right\rangle. \quad (2)$$

The first variant, *map/decompose* (MD) (Fig. 2: left path), directly encodes terms provided by $f$ into gates in the circuit and then starts a decomposition step. This comes with the downside of higher circuit depth and more gates in the circuit. The second variant, *reduce/map/decompose* (RMD) (Fig. 2: right path), order-reduces $f$ with $\deg(f) > 2$ to a quadratic PBF $f'$ (*i.e.*, a QUBO), maps terms from $f'$ to the circuit and then decomposes them to the hardware gate-set. A reduction introduces additional variables, and therefore increases the number of required qubits. To avoid burdening the comparison with hardware-specific details, we assume that the hardware gate-set consists of single-qubit $R_Z$, $R_X$ operations, and the two-qubit gate C–$X$.

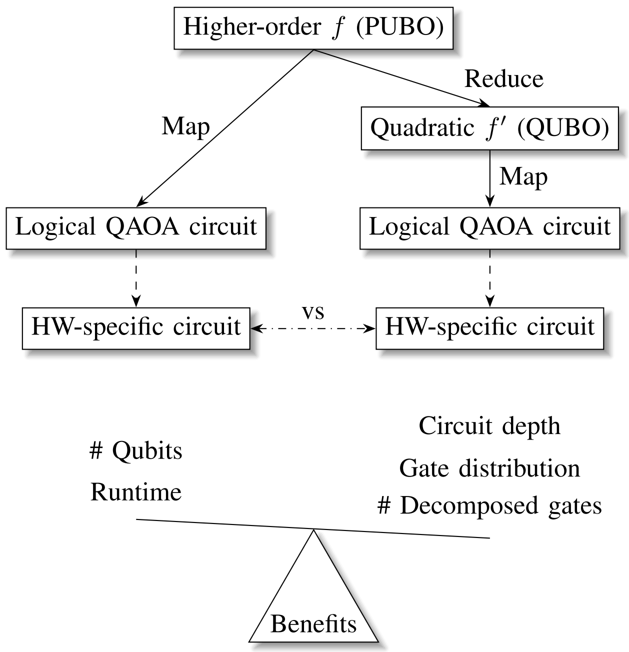

Figure 2: Overview of reduction strategies for higher-order unconstrained problems $f$ to QAOA, and their consequences for execution on quantum hardware. Dashed line: Decomposition part of the transpilation process.

This leads to our research questions:

RQ1 How do circuits based on higher-order functions originating from *map/decompose* (MD) and *reduce/map/decompose* (RMD) differ in (a) depth, (b) size, and (c) structure?

RQ2 What are the implications on the quantum software regarding abstraction from hardware specific peculiarities?

The rest of this paper is structured as follows: We first introduce the fundamental concept of *quadratisation* in Sec. II, followed by details on formal representation and properties of PBFs in Sec. III, which also introduces a graph representation to visualise PBFs and the effect of reductions. The experimental analysis in Sec. IV is based on a practical use case, a scheduling problem, to understand important performance metrics of quantum circuits for strategies MD and RMD. We conclude by discussing the impact of our findings on quantum software in Sec. V.

The paper is augmented by supplementary website and a comprehensive reproduction package [17] (links in PDF) that allows for extending our work.

## II. Related Work

Improving the performance of quantum algorithms is an intensively studied topic at many levels of abstraction. Published findings concern issues from high-level software engineering to low-level design of compilers [18]. Often, well established concepts from classical computing are evaluated in the quantum computing context. Ahmad *et al.* [19] investigate modular architecture centric designs. Scheerer *et al.* [20] evaluate architectural patterns for fault-tolerant systems. Faro *et al.* [21] and Saurabh *et al.* [22] propose techniques to integrate quantum computation into data centres by the means of existing system architectures. At a lower level, Codognet *et al.* [23] compare annealing(-inspired) approaches, and Schmale *et al.* [18] review compiler phases for trapped ion devices. Often, changes in one domain affect the performance in a lower domain. Guimares and Tavares [24] Safi *et al.* [25] and Wintersperger *et al.* [26] address the HW-SW co-design aspects of quantum computing, which tries to balance restrictions in the realisability of hardware and positive effects in the software domain, such as qubit connectivity or error proneness of NISQ devices.

In principle it is possible to realise interaction gates involving more than two qubits, for example, with trapped-ion quantum computers [27], [28]. However, this is currently not possible on other platforms such as superconducting qubits, and not provided by any commercial quantum devices. Nonetheless, higher order gates can be replaced by decomposing them into multiple smaller gates that lead to the same quantum operation. For QAOA, an interesting decomposition is concerned with rotation gates around the $Z$-axis. Suppose the single-qubit $R_Z$ gate can be implemented natively. Then the two-qubit $R_{ZZ} := R_{Z^2}$ gate can be decomposed into a symmetric arrangement of two C–$X$-gates around a single-qubit $R_Z$ gate. This method can be extrapolated to $R_{Z^n}$ gates and is motivated by the works of Campbell and Dahl [29], with reference to ZX-calculus [30]. They analyse the effect of decomposition for the four corner graph colouring problem. They were able to execute small instances of QAOA with COBYLA—a classical optimiser for $\vec{\beta}$ and $\vec{\gamma}$ (see Eq. 2)—and found better performance for the mere decomposition strategy (MD), which differs from our findings. They suggest that the use of higher order terms can be beneficial, which motivates our work.

In contrast to the MD strategy, one can also modify the OP so that higher-order gates are no longer needed. This is also interesting in view of using quantum annealing, as currently available devices can only handle interactions between a maximum of two qubits [14].

One particular approach is *quadratisation* [31], which eliminates the need for $R_{Z^n}, n > 2$ gates. A function $f'(\vec{x}, \vec{y})$ is a *quadratisation* of $f(\vec{x})$, if $f'(\vec{x}, \vec{y})$ is a quadratic ($\deg(f') = 2$) PBF in $\vec{x} = x_1, \ldots, x_n$ and $\vec{y} = y_1, \ldots, y_m$, and satisfies:

$$f(\vec{x}) = \min_{\vec{y} \in \{0,1\}^m} f'(\vec{x}, \vec{y}) \; \forall \vec{x} \in \{0,1\}^n. \tag{3}$$

Note that with this also the minimum over $\vec{x}$ is preserved.

A variety of methods, reviewed by Dattani [32], are known to construct a quadratisation. Typical approaches include rewriting $f(\vec{x})$ through logical reasoning of minima or exploiting substructures in $f(\vec{x})$ (*e.g.*, via *a priori* knowledge [33]; splitting the objective function [34]; excluding monomials [33]; partial monomial assignment [35] or using Gröbner basis techniques [36]). It is not uncommon that some methods require $f$ to satisfy particular properties (*e.g.*, restricted degree [37]; exclusively negative or positive monomials [38], [39] or non-binary variables [40]). A versatile method proposed by Boros [41] quadratises an *arbitrary* PBF $f : \{0,1\}^n \to \mathbb{R}$ constructively by iteratively replacing a pair of variables $x_i x_j$ in the multi-linear representation of $f$ by a new variable $y_h$. To ensure the constraint specified in Eq. 3 and enforce $y_h = x_i x_j$ in the minimisation of $f$, a penalty $\mathrm{p}$ is added to the reduced PBF, which fulfils

$$\begin{aligned} x_i x_j = y_h &\Rightarrow \mathrm{p} = 0 \\ x_i x_j \neq y_h &\Rightarrow \mathrm{p} > 0, \end{aligned} \tag{4}$$

where $\mathrm{p}$ is usually chosen as

$$\mathrm{p}(x_i, x_j, y_h) = 3y_h + x_i x_j - 2x_i y_h - 2x_j y_h. \tag{5}$$

## III. Fundamentals

### A. Pseudo Boolean Functions and their Graph Representation

*a) Pseudo Boolean Functions:* Following Boros *et al.* [31], a PBF $f : \{0,1\}^n \to \mathbb{R}$ can be expressed in an algebraic representation by multi-linear polynomials [41]

$$f(x_1, \ldots, x_n) = \sum_{S \subseteq \{1,\ldots,n\}} \alpha_S \prod_{j \in S} x_j, \tag{6}$$

where $\alpha_S \prod_{j \in S} x_j$ is called a monomial of $f$ and $\alpha_S \in \mathbb{R}$. For example, $f(x_1, x_2, x_3) = \pi x_1 + 3x_1 x_2 - 17 x_2 x_3$ is a PBF and $\pi x_1$, $3x_1 x_2$ and $-17 x_2 x_3$ are monomials of $f$.

In the following, we introduce terminology related to PBFs. Let $m_S$ be a monomial of the PBF $f$, where $S \subseteq \{1, \ldots, n\}$ is a subset of indices. Then, we define the degree of $f$ by the maximum degree over $f$'s present ($\alpha_S \neq 0$) monomials:

$$\deg(f) = \max_{S \subseteq \{1,\ldots,n\}, \alpha_S \neq 0} \deg(m_S), \tag{7}$$

where we define the degree of a monomial $m_S$ by the number of variables it contains:

$$\deg(m_S) = |S| \in \{0, \ldots, n\}. \tag{8}$$

Note that $x^k = x$ for all $k \in \mathbb{N}$ for all binary variables $x \in \{0,1\}$. Furthermore, we define the degree-$k$ ($k \in \mathbb{N}_0$) density of $f$ by the ratio $\frac{\text{actual}}{\text{possible}}$ degree-$k$ monomials. Since there are $\binom{n}{k}$ possible degree-$k$ monomials[1],

$$d_k = \frac{t_k}{\binom{n}{k}} \in [0, 1], \tag{9}$$

[1]For the construction of a degree-2 monomial, there are $n(n-1)$ variable combinations. For a degree-3 monomial, there are $n(n-1)(n-2)$ combinations, etc. Since multiplication is commutative in each monomial, permutations, of which there are $k!$ many, are irrelevant. Consequently, there are $\frac{n!}{(n-k)!k!} = \binom{n}{k}$ ways to construct a degree-$k$ monomial.

where $t_k$ denotes the number of actual terms of degree-$k$. For example, the function $f(x_1, x_2, x_3) = 3 + 1x_1x_2 - 2x_2x_3 + 7x_1x_2x_3$ has degree 3, since $\max\{\deg(m_{\{\}}), \deg(m_{\{1,2\}}), \deg(m_{\{2,3\}}), \deg(m_{\{1,2,3\}})\} = \max\{|\{\}|, |\{1,2\}|, |\{2,3\}|, |\{1,2,3\}|\} = 3$. Furthermore, the degree-$k$ densities of $f$ are as follows:

$$d_0 = \frac{1}{\binom{3}{0}} = 1,\ d_1 = \frac{0}{\binom{3}{1}} = 0, \\ d_2 = \frac{2}{\binom{3}{2}} = \frac{2}{3},\ d_3 = \frac{1}{\binom{3}{3}} = 1,\ d_{k>3} = 0. \tag{10}$$

*b) Graph Representation for Pseudo Boolean Functions:* An undirected graph $G(V, E)$ is, as usual, given by vertices $v_i \in V$ and its edges $e = \{v_i, v_j\} \in E$. We also consider multigraphs that can have duplicate edges so that $E$ becomes a multiset. With reference to Eq. 6, the set $V = \{v_1, \dots, v_n\}$ is isomorphic to the set of variables in $f$. While $x_i$, for $i \in \{1, \dots, n\}$, is a variable in $f$, $v_i$ represents that variable in $G$. The multiset of edges $E$ is constructed over $f$'s monomials. Let $S \subseteq \{1, \dots, n\}$ be a subset of indices and let

$$P_S = \{\{v_a, v_b\}|\ a \in S \wedge b \in S\ \wedge a \neq b \wedge \alpha_S \neq 0\} \tag{11}$$

be the two-combination set of $S$. Then $P_S$ is the set of edges generated by iterating over possible two-combinations of variables in a monomial specified by $S$. Then, $E$ is the multiset-sum over all $P_S$. For example, the graph $G(V, E)$ of

$$f(x_1, \dots, x_6) = 3x_1 - 2x_2x_3 + 5x_1x_2x_3x_6 - 2x_1x_2x_4x_5 \tag{12}$$

is defined by $V = \{v_1, \dots v_6\}$ and

$$\begin{aligned} E &= \sum_{S \subseteq \{1,\dots,6\}} P_S \\ &= P_{\{2,3\}} + P_{\{1,2,3,6\}} + P_{\{1,2,4,5\}} \\ &= \{\{v_2, v_3\}, \\ &\quad \{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_6\}, \{v_2, v_3\}, \{v_2, v_6\}, \{v_3, v_6\}, \\ &\quad \{v_1, v_2\}, \{v_1, v_4\}, \{v_1, v_5\}, \{v_2, v_4\}, \{v_2, v_5\}, \{v_4, v_5\}\}. \end{aligned}$$

This graph can be seen in Fig. 3, where its vertices are drawn as the corners of a regular hexagon. Here, multi-edges

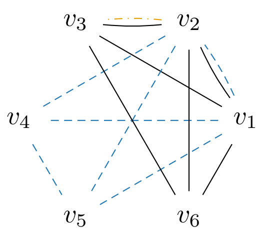

Figure 3: Multigraph of Eq. 12 before the reduction.

are drawn as separate edges in the graph, but can also be visualised by using an edge weight on a single edge. Hence, an equivalent notation of (multi-)edges uses exponential notation: $\{v_i, v_j\}^\beta$ is called a multi-edge iff $\beta > 1, \beta \in \mathbb{N}$.

Notice that the graph representation is concerned with the structure of $f$, but not the numerical influence of a monomial of $f$, that is, the value of $\alpha_S$. Let $G(V, E)$ be the graph corresponding to a PBF $f$. If $G$ has multi-edges, then $\deg(f) > 2$ and equivalently, if $\deg(f) \leq 2$, $G$ has no multi-edges. The inverse relation is not true in the general case. An example is $f(x_1, x_2, x_3) = x_1x_2x_3$. The corresponding graph $G$ is fully connected, but has no multi-edges, while $\deg(f) = 3$. Sec. III-C discusses the implications for the reduction process.

## B. Variants of Boros Reduction

Boros reduction [41] can iteratively reduce a higher order PBF $f$ to a quadratic one, while leaving the choice of the next variable pair during an iteration ambiguous (see Sec. II). Although any variable pair in a monomial of $f$ is valid, a sensible choice involves characteristics of $f$. We therefore evaluate the following three choices for the next variable:

1) *Sparse*: Gets the largest degree monomial. Chooses its first variable pair for the next iteration step
2) *Medium*: Gets all monomials with largest degree. Searches for the variable pair that appears most often.
3) *Dense*: Searches for the variable pair that appears most among all monomials.

The open source framework *quark* (see Ref [42]) implements these methods. They influence the reduction process and its outcome to the point where knowledge about the hardware topology can be used in advance to adjust the reduction process to optimise the hardware specific quantum circuit. The variants are named according to the properties they induce in the resulting QUBO.

## C. The Influence of Reductions

Recall that Boros reduction introduces a penalty term in every iteration: $\mathrm{p}(x_i, x_j, y_h) = 3y_h + x_ix_j - 2x_iy_h - 2x_jy_h$.

*a) The Effect on multi-linear Polynomials:* Let $f : \{0,1\}^n \to \mathbb{R}$ be a PBF, such that $\deg(f) > 2$. A reduction results in a quadratic PBF $f'$: $\deg(f') = 2$. During the reduction, the new variables $y_1, \dots, y_m$ are introduced, which adapt the domain of $f'$:

$$f' : \{0,1\}^{n+m} \to \mathbb{R}. \tag{13}$$

Furthermore, the reduction process preserves the minimum of $f$ in $f'$, which follows from the quadratisation process of Eq. 3. This property lays the ground to the use of reductions in the reformulation of PBFs. Take into consideration that it might be necessary to scale the added term $\mathrm{p}$ with a large positive factor when solving $f'$ such that the penalty for breaking the constraint outweighs possible benefits in the objective function. This, however, does not influence the graph structure.

*b) The Effect on the Graph Representation:* Let $f$ be a PBF, such that its graph $G(V, E)$ contains multi-edges. For example, Fig. 4 shows such a function. The multi-edge $\{v_0, v_1\}^4$ is replaced by a single edge (from the penalty term) in the first step. Additionally, the penalty term connects the new node $y_1$ to the former nodes $v_0$ and $v_1$. Due to $x_0x_1$

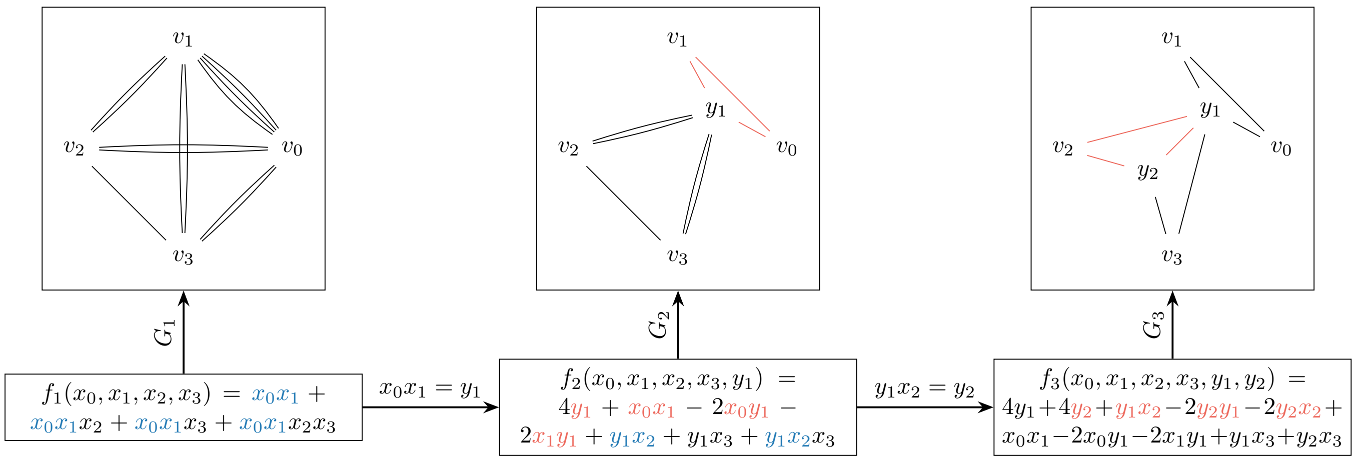
$v_1$
$v_2$
$v_0$
$v_3$
$v_1$
$y_1$
$v_2$
$v_0$
$v_3$
$v_1$
$y_1$
$v_2$
$y_2$
$v_0$
$v_3$
$G_1$
$G_2$
$G_3$
$f_1(x_0,x_1,x_2,x_3) = x_0x_1 + x_0x_1x_2 + x_0x_1x_3 + x_0x_1x_2x_3$
$x_0x_1 = y_1$
$f_2(x_0,x_1,x_2,x_3,y_1) = 4y_1 + x_0x_1 - 2x_0y_1 - 2x_1y_1 + y_1x_2 + y_1x_3 + y_1x_2x_3$
$y_1x_2 = y_2$
$f_3(x_0,x_1,x_2,x_3,y_1,y_2) = 4y_1 + 4y_2 + y_1x_2 - 2y_2y_1 - 2y_2x_2 + x_0x_1 - 2x_0y_1 - 2x_1y_1 + y_1x_3 + y_2x_3$

Figure 4: A complete multi reduction evolution. Blue symbols represent the variable pair to be replaced; orange lines and symbols represent edges and terms introduced via the penalty term in the current step. All variable-selection types (*i.e.*, *Sparse*, *Medium*, *Dense*) lead to identical results, assuming multiple valid choices are resolved by choosing the first pair for each type.

being replaced in every monomial by $y_1$, edges from these monomials are now mapped to $y_1$.

More generally, suppose the variable selection type chooses the variable pair $x_ix_j$ and therefore the multi-edge $\{v_i, v_j\}^\beta$ in a PBF $f_\gamma : \{0,1\}^n \to \mathbb{R}$ and its corresponding graph $G_\gamma(V_\gamma, E_\gamma)$ for iteration step $\gamma$. The multi-edge $\{v_i, v_j\}^\beta$ is replaced by a single edge $\{v_i, v_j\}$, as every occurrence of $x_ix_j$ is replaced by $y_\gamma$ apart from the term introduced in the penalty term itself. Additionally, the penalty term introduces $\{v_i, y_\gamma\}, \{v_j, y_\gamma\}$. Let $m$ denote a monomial in $f_\gamma$. Either (a) $x_ix_j \in m$ or (b) $x_ix_j \notin m$. Let $I = \{\{v_\tau, v_i\} \in E_\gamma | \ \tau \in \{1,\ldots,n\} \setminus \{j\}\}$ be the subset of edges connected to $v_i$ - excluding the ones connecting $v_i$ and $v_j$. Analogously, let $J = \{\{v_\tau, v_j\} \in E_\gamma | \ \tau \in \{1,\ldots,n\} \setminus \{i\}\}$ be the subset of edges connected to $v_j$ - excluding the ones connecting $v_i$ and $v_j$. Every edge $e \in I \cup J$ stemming from (a) in $G_\gamma$ will be reconnected to $y_\gamma$ in $G_{\gamma+1}$. Conversely, every edge $e \in I \cup J$ stemming from (b) in $G_\gamma$ is invariant in $G_{\gamma+1}$. Fig. 5 visualises the above stated, while the penalty term induced edges are coloured in orange. The remaining edges, that is the ones not connected to $v_i$ or $v_j$ in $G_\gamma$, are invariant in $G_{\gamma+1}$. This locality can be used for parallel execution of multiple reduction steps and thus aids in the speedup necessary for efficiently deploying the RMD strategy. Apart from parallel execution it lays the ground for an efficient data structure that does not need to reiterate over all monomials in search for the next variable pair in each reduction step.

An edge in the graph corresponds to either a degree-2 monomial or a higher degree monomial. Every edge, corresponding to a higher degree monomial is a potential candidate for the reduction. The *Dense* variable selection type restricts its choice to variable pairs that occur most often among all monomials - or equivalently the biggest multi-edges. The *Sparse* and *Medium* variable selection types lift that constraint and thus might select other edges. This is the only difference, yet it leads to vastly different properties of the resulting QUBO. We

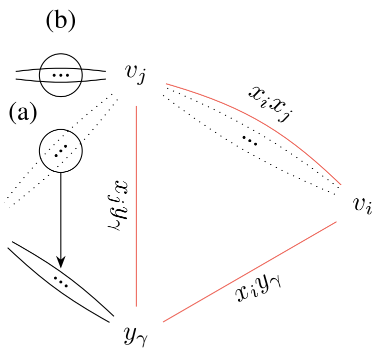

Figure 5: Effect of a reduction on $v_i$ and $v_j$. Dotted lines: Removed edges. Orange lines: Edges introduced by the penalty term. $\{v_i, v_j\}^\beta$ corresponds to $x_ix_j$, which is replaced by $y_\gamma$. The remapping of (a) ($x_ix_j \in m$) and the invariance of (b) ($x_ix_j \notin m$) is analogous for $v_i$ (not drawn).

observe differences in the edge distribution on nodes among the three variable selection variants concerning the graph representation. While the *Sparse* method concentrates its edges among the former variables, the *Dense* method distributes its edges among all nodes, including newly introduced ones. The *Medium* method lies in between these two. Fig. 6 illustrates the graph representation for a PBF $f$ ($deg(f) = 4$), in the top left corner and shows the graph for the resulting quadratic PBF $f'$ for each variable-selection type. The graph of a quadratic PBF $f'$ is a representation of its QUBO (except for linear and constant terms), and thus describes coupling structure and density, since the pair-combination sets $P_S$ of $f'$ are pairwise disjoint. Therefore, the properties of $f'$ in Fig. 6 characterise the structure of the QUBO. Sec. IV discusses further differences that are relevant for our performance measurements.

Reductions leave some graph properties invariant:

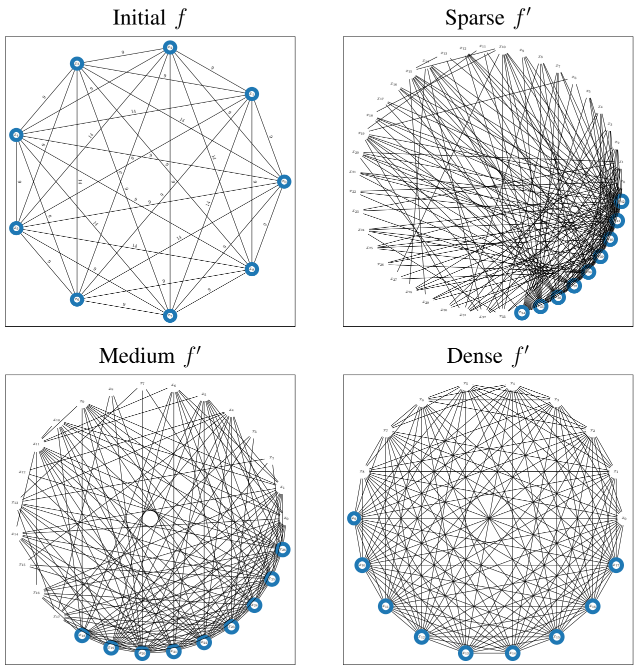

Figure 6: A complete multi reduction evolution in terms of the graph structure. The symmetric graph in the top left corner shows the initial PBF $f$ ($\deg(f) = 4$). The other nodes show a particular variant of Boros [41] reduction method starting from the initial PBF. Nodes circled in blue represent the initial variables.

1) The total size of multi-edges in the graph strictly decreases.
2) The degree of any node in the graph does not increase (excluding the new node in each iteration).

While we defer formal proofs to the supplementary website, note that they provide insights into the reductions: Usually, the starting PBF $f := f_0$ ($\deg(f_0) > 2$) corresponds to a graph with multi-edges [2]. When using the *Dense* variable-selection method, the algorithm at first reduces $f_0$ to $f_t$ after $t \in \mathbb{N}$ reduction iterations. The graph corresponding to $f_t$ might not contain multi-edges anymore, however $f_t$ might still not be a quadratisation of $f_0$. Hence, the algorithm enters stage two, where it can no longer select multi-edges and is therefore less efficient in terms of introduced variables, since it can no longer replace the same variable pair in multiple monomials. We can calculate the amount of introduced variables in stage two for $f_t$ by evaluating each monomial. A prerequisite of entering stage two is that the sets $P_S$ for $f_t$ are pairwise disjoint. It is easy to see that for any monomial $m$ with $\deg(m) = k > 2$, we require $k - 2$ new variables to reduce it to degree 2. Take for example the monomial $m = x_1x_2x_3$. Here, one extra variable is needed for the quadratisation of $m$ regardless of the choice of the variable pair.

[2]PBFs with pairwise disjoint sets $P_S$ do not induce multi-edges in their graph representation; for example, this is the case for $f(x_1, \ldots, x_n) = x_1x_2x_3 + x_4x_5x_6 + \ldots + x_{n-2}x_{n-1}x_n$.

## IV. Experiments

### A. Experimental Setup

Our experiments evaluate a PBF $f$ that encodes higher-order terms and can scale in the number of variables. It is motivated by an industrial job-shop scheduling problem: Production jobs must be assigned to machines such that the total runtime of a factory (the makespan) is minimised. For each job, only a single operation can be performed, and all machines are of the same type, so each job can in principle be run on each machine. The jobs require certain tool setups and are divided into multiple setup groups. Changing the setup takes additional time, which constrains optimal assignment. Here, we only aim for the allocation of jobs to machines, and leave the exact ordering jobs to classical post-processing. The problem is modelled as a PUBO using binary variables $x_{ij}$ with

$$x_{ij} = \begin{cases} 1 & \text{if job } i \text{ is assigned to machine } j, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

This yields a problem size of $n = N \cdot M$ qubits for $N$ jobs and $M$ machines. The goal is to minimise the maximum runtime including setup change times over all machines, which is expressed as minimising the differences between runtimes over all pairs of machines. Normally, there are many more jobs than machines and the job durations $d_i$ are longer than the setup change times. The setup change times between two jobs $i$ and $i'$ are given by the entries of the $N \times N$ matrix $R_{ii'}$, while another $N \times M$ matrix $S_{ij}$ describes the setup change time between a job $i$ and the initial setup of machine $j$. Using the variables from Eq. 14, the objective term is given by

$$H_{\text{obj}} = \sum_{j<j'} \Bigg[ \Bigg( \sum_{i=1}^{N} x_{ij}(d_i + S_{ij}) + \sum_{i<i'} x_{ij}x_{i'j}R_{ii'} \Bigg) - \Bigg( \sum_{i=1}^{N} x_{ij'}(d_i + S_{ij'}) + \sum_{i<i'} x_{ij'}x_{i'j'}R_{ii'} \Bigg) \Bigg]^2 . \tag{15}$$

As the order of jobs is not considered, we evaluate the maximum number of setup changes here. To avoid that the runtimes of two machines are equalised by adding additional, unwanted setup changes, we add a second term that minimises the setup changes on each machine separately:

$$H_r = \sum_{j=1}^{M} \Bigg[ \sum_{i<i'}^{N} x_{ij}x_{i'j}R_{ii'} + \sum_{i=1}^{N} x_{ij}S_{ij} \Bigg] . \tag{16}$$

Moreover, each job has to be assigned exactly once which is described by the following constraint:

$$H_{\text{single}} = \sum_{i=1}^{N} \Bigg( \sum_{j=1}^{M} x_{ij} - 1 \Bigg)^2 . \tag{17}$$

The PBF $f : \{0,1\}^n \to \mathbb{R}$ ($\deg(f) = 4$) arises naturally from these preliminary steps as

$$f = AH_{\text{obj}} + BH_r + CH_{\text{single}}, \tag{18}$$

where $C > A, B$ must hold to ensure valid solutions are preferred. We use $A = 1$, $B = 2 \cdot \max_i d_i$, $C = 4 \cdot \max_i d_i^2$ and choose $R$ and $S$ to have full rank.

For each of Fig. 7, 8 and 9, the x axis shows the number of variables before the reduction of $f$, that is, problem size $n$. Dependent on this characteristic, we evaluate run-time and variable overhead of the proposed reduction variants. Additionally, we compare number of introduced gates, circuit depth and polynomial structure for both paths of MD and RMD. For this, we create the problem-specific part of the QAOA circuit for a single layer without mixer. To compare both paths, we decompose $R_{Z^n}$, $n \geq 2$ gates into their symmetric representation given by Campbell and Dahl [29]. Here, we do not optimise for circuit depth. While this influences the comparison of circuit depths, we note optimising the depth of a quantum circuit is in itself $\mathbf{NP}$-complete [43].

## B. Experimental Results

Fig. 7 (bottom) shows the runtime for each variable-selection type of Boros [41] reduction method. While the

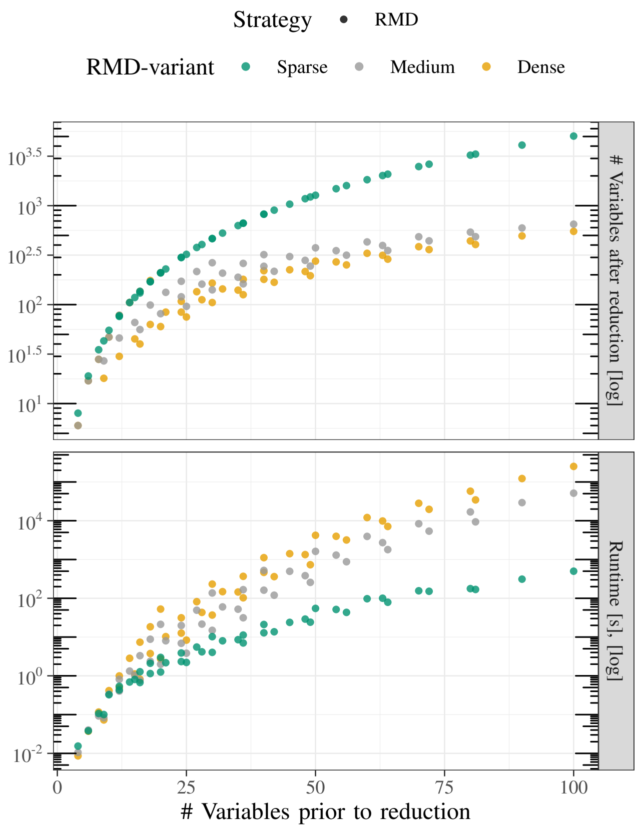

Figure 7: Top: Number of qubits (*i.e.* variables) after the reduction over problem size as captured by the number of variables before the reduction used to assess the spatial feasibility (*reduce/map/decompose* (RMD) strategy). Bottom: Runtime (pure reduction) over problem size and reduction type for the RMD strategy to assess computational feasibility.

total number of steps for a reduction, that is the number of introduced variables, can be estimated using $f$'s multi-graph (see Sec. III-C), the total runtime also depends on the search and replacement implementation. The latter one is shared among the variable-selection variants. Fig. 7 (bottom) shows a diverging runtime trajectory between the *Sparse* and *Dense* variable-selection type. This traces back to the search strategy as the dominant part for the next variable selection type. The *Sparse* variant introduces more variables, as Fig. 7 (top) illustrates, which means that the replacement occurs more often, which increases the runtime[3] on the one hand. On the other hand, the *Sparse* variant has a far less computationally intensive search strategy, which results in an overall significantly lower runtime, compared to the *Dense* variant. Since the *Medium* selection-type is computationally less expensive than the *Dense* type concerning the search strategy, we expect a lower runtime. Fig. 7 confirms this.

Apart from the computational feasibility of the reductions, their spacial influence on PBFs is relevant to quantum computing, as the number of variables translates to the number of qubits in the quantum circuit. Fig. 7 (top) shows the dependency of the number of variables after the reduction (y-axis) on the the number of variables before the reduction (x-axis) grouped by the variable-selection type. For the absolute values, the underlying polynomial and its structure is decisive, since it determines the size and coupling density of the corresponding multi-graph $G(V, E)$. In contrast to the runtime, the variable-overhead mirrors the roles of the *Sparse*, *Medium* and *Dense* types. While the *Sparse* type took the least amount of time, it introduces significantly more variables. The reason behind this is the less efficient selection type, which chooses smaller (multi-)edges on average in the graph representation during the reduction process. The upper bound for the number of introduced variables during a reduction is given by the polynomial's degree. To be more precise, let $f : \{0, 1\}^n \to \mathbb{R}$ be the function of interest, such that $\deg(f) = i$. There are at most $\binom{n}{k} = \mathcal{O}(n^k)$ degree-$k$ monomials (see Sec. III-A). At maximum, one needs $k - 2$ extra variables per degree-$k$ monomial (see Sec. III-C) and thus $\mathcal{O}(\sum_{k=3}^{i} kn^k) = \mathcal{O}(n^i)$ extra variables for the quadratisation of $f$. In our case, that is $\deg(f) = 4$, the number of introduced variables is upper bounded by $\mathcal{O}(n^4)$. Note that Fig. 7 (top) shows the sum of the number of introduced variables and variables before the reduction on the y-axis (*i.e.*, $n + m$; see Eq. 13).

We now want to characterise the underlying function $f$ in terms of densities to get a better understanding of the above stated upper bound and to anticipate the influence on the quantum circuit. Fig. 8 shows the degree-$k$ densities $d_k$, as introduced in Sec. III-A, of $f$ before the reduction (MD) and of its quadratic counterpart $f'$ grouped by the variable-selection type (RMD). For the quadratic PBF $f'$, $d_{k>2} = 0$, since there are no monomials $m$ in $f'$, such that $\deg(m) > 2$. The density $d_1$ does not change in this case, since $f$ incorporates

[3]The replacement occurs on different polynomials, which affects the runtime. We estimate this effect to be marginal for the given implementation.

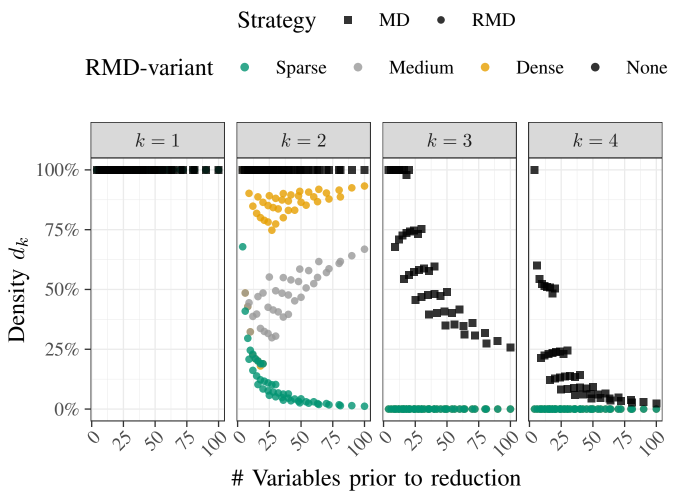

Figure 8: Polynomial density over problem size as measured by the number of variables prior to reduction. The graph shows the reduction from a higher order polynomial (MD) to a quadratic PBF (RMD) for different variable selection types.

the maximum number of degree-1 terms already. In each reduction step, a new variable is introduced. At the same time, a unique degree-1 term is added to $f$ through the penalty term (see Sec. III-A). For the general case, as the number of reduction steps increases, $d_1$ converges to the maximum value, that is $d_1 \to 1$, regardless of the number of degree-1 terms in the original function[4]. Interestingly, the convergence behaviour of $d_2$ is dependent on the variable-selection type. Fig. 8 shows this through our quantitative analysis. While the *Dense* selection type seems to converge to $d_2 > 90\%$, the *Sparse* type converges to $d_2 < 10\%$. Similar to Fig. 7, the *Medium* type lies in between the *Sparse* and *Dense* type. This has immediate consequences for the quantum circuit, since there are more variables in case of the *Sparse* method, but less two-qubit interactions between them, as indicated by the lower density. As not all possible pairs interact, the two-qubit gates can be parallelised more easily. Take into consideration that the underlying function $f$ does not feature every degree-4 and degree-3 monomial. In fact, the densities $d_3$ and $d_4$ converge to zero, which is caused by the choice of $f$. We estimate, that our findings for the convergence hold for similar structured functions as well. Recall the definition of $d_2$ from Eq. 9. Since

$$\lim_{n\to\infty} d_2 = \lim_{n\to\infty} \frac{t_2}{\Theta(n^2)} = \begin{cases} 0 & \text{if } t_2 \in o(n^2) \\ c & \text{if } t_2 \in \Theta(n^2) \end{cases}, \tag{19}$$

such that $0 < c \leq 1$, we argue that, based on Fig. 8, the *Dense* selection type is efficient in terms of encoding information.[5]

[4]The rationale being $\lim_{i\to\infty} \frac{t_1+i}{\binom{n+i}{1}} = 1 \; \forall t_1 \in \{0, \ldots, n\}$.

[5]This cannot be traced back to the penalty term, since it adds exactly 3 edges per iteration (see Sec. III-C). It is therefore a linear function in terms of the iteration count and does not close the gap to the quadratic function.

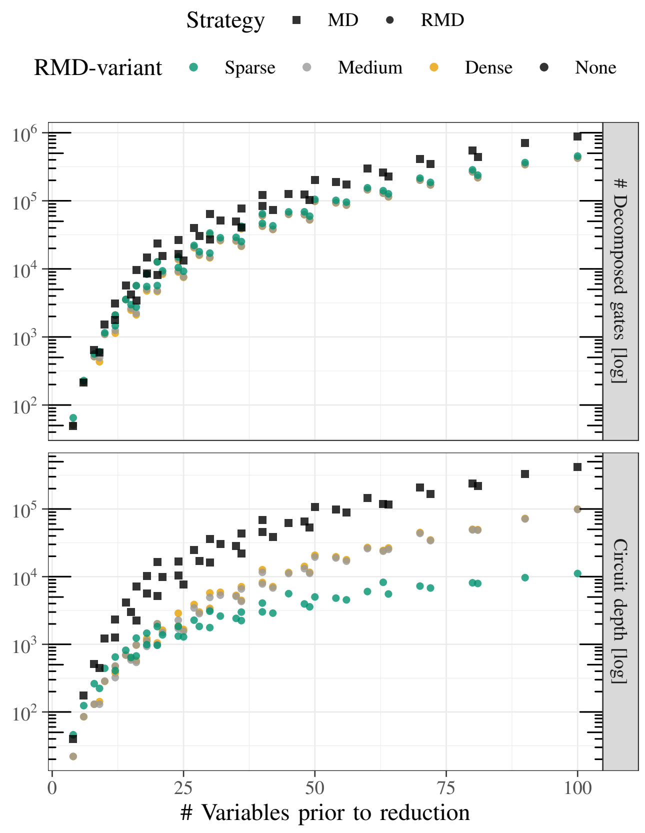

Figure 9: Single- and two-qubit gates (excluding gates that implement the mixer operation) in one QAOA layer over problem size as captured by the the number of variables for strategies *map/decompose* (MD) and *reduce/map/decompose* (RMD). Top: Gate count used as proxy for gate errors; bottom: Circuit depth (not optimised; gates inserted lexicographical) as proxy for decoherence effects.

The top part of Fig. 9 classifies the variable-selection types based on the number of introduced gates in a single QAOA layer without the mixer. The variable-selection types encode approximately the same information in their respective quadratic polynomials, since they introduce approximately the same number of gates. In terms of the number of introduced gates in a QAOA circuit, the RMD strategy is superior to the MD strategy. Take into consideration, the decomposition of higher-order gates influences this metric. The number of gates is an important measure in the design of quantum circuits, as the cumulative effect of gate-errors invalidates the quantum state in the NISQ-era.

In contrast to the bare number of gates, the circuit depth takes into account possible parallel execution of gates. It determines the quantum algorithm's runtime and therefore the extent of decoherence effects. The bottom part of Fig. 9 visualises circuit depth, where the RMD strategy features a lower circuit depth than the MD strategy. In short, the RMD strategy suffers from more variables, while at the same time encoding less gates in the circuit, which results in far shorter circuits. Although the *Medium* type introduces more variables compared to the *Dense* type (see Fig. 7 top), it features approximately the same circuit depth at bigger problem instances. Take into consideration that we do not optimise the circuit depth here. Apart from that, the circuit depth is considerably lower for the *Sparse* type, since it introduces considerably more variables.

## V. Implications on Quantum Software

In modern software architectures, abstraction layers strive to conceal details about lower layers to allow software engineers to concentrate their work on the relevant architectural state. In essence, the *SWE/Modelling Domain* and the *Tool/Compiler Domain* are coupled through non-functional requirements, originating from both domains. It is important to analyse the effect of decisions in one domain to other abstraction layers. One could argue that more abstraction layers benefit software engineering. However, as Schönberger *et al.* [44] showed, this is not necessarily the case for quantum software. Furthermore, abstraction layers need to be specified without hiding relevant information. For instance, the runtime limitation of (quantum) algorithms is a cross-abstraction-layer non-functional requirement, for which we showed a substantial increase, when using the RMD strategy. On the other hand, other non-functional requirements are positively influenced by the RMD strategy. Hence, the best fitting balance is to be found, while considering both upstream and downstream influences of decisions in abstraction layers.

By incorporating hardware specifications in advance, we can simultaneously lift constraints about the formulation of OPs (*i.e.*, using PUBOs) and optimise for hardware efficient execution. For this we compared two strategies, namely *map/decompose* (MD) and *reduce/map/decompose* (RMD) with regard to their impact on important metrics of quantum circuits.

Our findings suggest that the use of polynomial order-reduction methods benefits the performance of quantum algorithms, provided that the hardware incorporates sufficiently many qubits. Therefore, the QUBO model, which translates to two-qubit interactions, is a non-critical aspect of modelling OPs from a software engineering perspective. Thus, the use of higher-order terms to model complex relations is possible, while simultaneously optimising for non-local properties of quantum circuits, namely the circuit depth and the number of gates. In contrast to that, the mere decomposition of gates, provides a local hardware specific remodelling. The circuit depth and the number of gates in a circuit are an indicator of the circuit's performance in the NISQ-era, where decoherence effects and gate errors invalidate the quantum state.

Using variant *Dense* to deploy higher-order OPs to fully connected quantum hardware such as trapped ion devices [45] achieves relatively low qubit overhead. For hardware with lower coupling densities, such as the heavy-hex topology [46], the *Sparse* or *Medium* type provide lower densities in the reduced polynomial, while using more qubits. We also see that the *Sparse* variant focuses its edges among the former variables, which benefits in-homogeneous hardware topologies with concentrated density regions, which also appear in quantum annealing devices [14]. The mapping and routing of logical quantum circuits to concrete hardware is an $\mathbf{NP}$-complete problem and is thus usually approached with approximation techniques [30], [47]–[50]. Selecting a particular reduction variant to suit the hardware topology can therefore aid in the approximation. However, further research is needed to assess this influence quantitatively.

## VI. Conclusion and Outlook

We have discussed that the high-level domain of software engineering and modelling activities is coupled, through non-functional requirements, with the low-level domains of compilers, tool-chains and quantum hardware. This complicates finding appropriate abstraction layers. By giving an in-depth analysis of a family of automated transformations from a relatively high-level description of optimisation problems in PUBO form to circuits for specific quantum hardware, we have shown that the process allows for exercising control over various non-functional properties of the resulting computation.

As determined by a numerical experimental analysis, the RMD strategy provides benefits in terms of quantum circuit depth, size and structure, as long as the underlying hardware offers a sufficient amount of qubits for the problem at hand. As our analysis only depends on the PBF's structure, the results can be extrapolated to similarly structured PBFs.

We provide points of reference that the *Dense* variable-selection type for the RMD strategy offers polynomial densities for its degree-2 terms converging to 1, whereas they converge to 0 for the *Sparse* type. By mapping these different quadratic polynomials (*i.e.*, QUBOs), resulting from the problem's PUBO, to a QAOA circuit, they influence its depth, size and structure decisively. As an outlook, we are not limited to the these variable selection types, since the reduction process is of iterative nature. As both types can be arbitrarily mixed during a reduction process, this allows us to achieve essentially arbitrary degree-2 densities in the resulting polynomial and thus a variety of opportunities to aid in the approximating of the otherwise $\mathbf{NP}$-complete mapping problem [47].

Using reduction methods in an online algorithm demands a higher level of algorithmic optimisation, especially for the *Dense* variable-selection type. This is to ensure that possible quantum advantage of the resulting circuit is not compensated by excessive preparatory classical effort (nevertheless, it is worth pointing out that the runtime scales in polynomial time for each selection type). Through the introduction and mathematical analysis of a graph structure, all selection types suggest potential for parallel execution of the transformation, which is a result of the local influence of a reduction step.

**Acknowledgements** We acknowledge support from German Federal Ministry of Education and Research (BMBF), funding program "Quantum Technologies—from Basic Research to Market", grant #13N15647 and #13NI6092 (LS, WM) and #13N16093 (KW). WM acknowledges support by the High-Tech Agenda Bavaria.

## References

[1] L. K. Grover, "A fast quantum mechanical algorithm for database search," 1996. [Online]. Available: https://arxiv.org/abs/quant-ph/9605043

[2] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Rev. Mod. Phys.*, vol. 90, p. 015002, Jan 2018. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.90.015002

[3] N. Pirnay, V. Ulitzsch, F. Wilde, J. Eisert, and J.-P. Seifert, "An in-principle super-polynomial quantum advantage for approximating combinatorial optimization problems via computational learning theory," *Science Advances*, vol. 10, no. 11, p. eadj5170, 2024. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.adj5170

[4] M. Schönberger, I. Trummer, and W. Mauerer, "Quantum-inspired digital annealing for join ordering," *Proc. VLDB Endow.*, vol. 17, no. 3, p. 511–524, nov 2023. [Online]. Available: https://doi.org/10.14778/3632093.3632112

[5] M. Schönberger, I. Trummer, and W. Mauerer, "Quantum optimisation of general join trees," in *Proceedings of the International Workshop on Quantum Data Science and Management*, ser. QDSM@VLDB, 2023.

[6] M. Schönberger, S. Scherzinger, and W. Mauerer, "Ready to leap (by co-design)? join order optimisation on quantum hardware," in *Proceedings of ACM SIGMOD/PODS International Conference on Management of Data*, 2023.

[7] T. Krüger and W. Mauerer, "Quantum annealing-based software components: An experimental case study with SAT solving," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ser. ICSEW'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 445–450. [Online]. Available: https://doi.org/10.1145/3387940.3391472

[8] F. Greiwe, T. Krüger, and W. Mauerer, "Effects of imperfections on quantum algorithms: A software engineering perspective," in *2023 IEEE International Conference on Quantum Software (QSW)*, 2023, pp. 31–42. [Online]. Available: https://doi.org/10.1109/QSW59989.2023.00014

[9] M. Franz, L. Wolf, M. Periyasamy, C. Ufrecht, D. Scherer *et al.*, "Uncovering instabilities in variational-quantum deep q-networks," *Journal of The Franklin Institute*, 8 2022.

[10] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014.

[11] Z. Bian, F. A. Chudak, W. G. Macready, and G. Rose, "The ising model : teaching an old problem new tricks," 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:15182277

[12] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023023, Feb. 2016. [Online]. Available: http://dx.doi.org/10.1088/1367-2630/18/2/023023

[13] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia *et al.*, "The variational quantum eigensolver: A review of methods and best practices," *Physics Reports*, vol. 986, pp. 1–128, 2022, the Variational Quantum Eigensolver: a review of methods and best practices. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0370157322003118

[14] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, "Perspectives of quantum annealing: methods and implementations," *Reports on Progress in Physics*, vol. 83, no. 5, p. 054401, May 2020. [Online]. Available: http://dx.doi.org/10.1088/1361-6633/ab85b8

[15] I. Sax, S. Feld, S. Zielinski, T. Gabor, C. Linnhoff-Popien *et al.*, "Approximate approximation on a quantum annealer," in *Proceedings of the 17th ACM International Conference on Computing Frontiers*. New York, NY, USA: Association for Computing Machinery, 2020, p. 108–117. [Online]. Available: https://arxiv.org/pdf/2004.09267

[16] S. Hadfield, Z. Wang, B. O'Gorman, E. Rieffel, D. Venturelli *et al.*, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, p. 34, Feb. 2019. [Online]. Available: http://dx.doi.org/10.3390/a12020034

[17] W. Mauerer and S. Scherzinger, "1-2-3 reproducibility for quantum software experiments," in *IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2022, pp. 1247–1248.

[18] T. Schmale, B. Temesi, A. Baishya, N. Pulido-Mateo, L. Krinner *et al.*, "Backend compiler phases for trapped-ion quantum computers," in *2022 IEEE International Conference on Quantum Software (QSW)*. IEEE, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1109/QSW55613.2022.00020

[19] A. Ahmad, A. A. Khan, M. Waseem, M. Fahmideh, and T. Mikkonen, "Towards process centered architecting for quantum software systems," in *2022 IEEE International Conference on Quantum Software (QSW)*. IEEE, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1109/QSW55613.2022.00019

[20] M. Scheerer, J. Klamroth, and O. Denninger, "Fault-tolerant hybrid quantum software systems," in *2022 IEEE International Conference on Quantum Software (QSW)*. IEEE, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1109/QSW55613.2022.00023

[21] I. Faro, I. Sitdikov, D. G. Valiñas, F. J. M. Fernandez, C. Codella *et al.*, "Middleware for quantum: An orchestration of hybrid quantum-classical systems," in *2023 IEEE International Conference on Quantum Software (QSW)*. IEEE, Jul. 2023. [Online]. Available: http://dx.doi.org/10.1109/QSW59989.2023.00011

[22] N. Saurabh, S. Jha, and A. Luckow, "A conceptual architecture for a quantum-hpc middleware," in *2023 IEEE International Conference on Quantum Software (QSW)*. IEEE, Jul. 2023. [Online]. Available: http://dx.doi.org/10.1109/QSW59989.2023.00023

[23] P. Codognet, D. Diaz, and S. Abreu, "Quantum and digital annealing for the quadratic assignment problem," in *2022 IEEE International Conference on Quantum Software (QSW)*. IEEE, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1109/QSW55613.2022.00016

[24] J. D. Guimaraes and C. Tavares, "Towards a layered architecture for error mitigation in quantum computation," in *2022 IEEE International Conference on Quantum Software (QSW)*. IEEE, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1109/QSW55613.2022.00022

[25] H. Safi, K. Wintersperger, and W. Mauerer, "Influence of HW-SW-co-design on quantum computing scalability," in *2023 IEEE International Conference on Quantum Software (QSW)*. IEEE, Jul. 2023. [Online]. Available: http://dx.doi.org/10.1109/QSW59989.2023.00022

[26] K. Wintersperger, H. Safi, and W. Mauerer, "Qpu-system co-design for quantum hpc accelerators," in *Architecture of Computing Systems*, M. Schulz, C. Trinitis, N. Papadopoulou, and T. Pionteck, Eds. Cham: Springer International Publishing, 2022, pp. 100–114.

[27] T. Monz, K. Kim, W. Hänsel, M. Riebe, A. S. Villar *et al.*, "Realization of the quantum toffoli gate with trapped ions," *Phys. Rev. Lett.*, vol. 102, p. 040501, 1 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.102.040501

[28] Y. Shapira, R. Shaniv, T. Manovitz, N. Akerman, L. Peleg *et al.*, "Theory of robust multiqubit nonadiabatic gates for trapped ions," *Phys. Rev. A*, vol. 101, p. 032330, 3 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.101.032330

[29] C. Campbell and E. Dahl, "QAOA of the highest order," in *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, 2022, pp. 141–146.

[30] A. Cowtan, S. Dilkes, R. Duncan, W. Simmons, and S. Sivarajah, "Phase gadget synthesis for shallow circuits," 2019. [Online]. Available: https://arxiv.org/abs/1906.01734

[31] E. Boros, Y. Crama, and E. Rodríguez-Heck, "Compact quadratizations for pseudo-boolean functions," *Journal of Combinatorial Optimization*, vol. 39, no. 3, pp. 687–707, 2019. [Online]. Available: https://doi.org/10.1007%2Fs10878-019-00511-0

[32] N. Dattani, "Quadratization in discrete optimization and quantum mechanics," 2019. [Online]. Available: https://arxiv.org/abs/1901.04405

[33] R. Tanburn, E. Okada, and N. Dattani, "Reducing multi-qubit interactions in adiabatic quantum computation without adding auxiliary qubits. part 1: The "deduc-reduc" method and its application to quantum factorization of numbers," 2015.

[34] E. Okada, R. Tanburn, and N. S. Dattani, "Reducing multi-qubit interactions in adiabatic quantum computation without adding auxiliary qubits. part 2: The "split-reduc" method and its application to quantum determination of ramsey numbers," 2015.

[35] H. Ishikawa, "Higher-order clique reduction without auxiliary variables," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1362–1369.

[36] R. Dridi and H. Alghassi, "Prime factorization using quantum annealing and computational algebraic geometry," *Scientific Reports*, vol. 7, no. 1, Feb. 2017. [Online]. Available: http://dx.doi.org/10.1038/srep43048

[37] A. C. Gallagher, D. Batra, and D. Parikh, "Inference for order reduction in markov random fields," in *CVPR 2011*, 2011, pp. 1857–1864.

[38] M. Anthony, E. Boros, Y. Crama, and A. Gruber, "Quadratic reformulations of nonlinear binary optimization problems," *Mathematical Programming*, vol. 162, no. 1–2, p. 115–144, Jun. 2016. [Online]. Available: http://dx.doi.org/10.1007/s10107-016-1032-4

[39] E. Boros and A. Gruber, "On quadratization of pseudo-boolean functions," 2014.

[40] A. Rocchetto, S. C. Benjamin, and Y. Li, “Stabilizers as a design tool for new forms of the lechner-hauke-zoller annealer,” *Science Advances*, vol. 2, no. 10, Oct. 2016. [Online]. Available: http://dx.doi.org/10.1126/sciadv.1601246

[41] E. Boros and P. L. Hammer, “Pseudo-boolean optimization,” *Discrete Applied Mathematics*, vol. 123, no. 1, pp. 155–225, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166218X01003419

[42] E. Lobe, “quark: Quantum application reformulation kernel,” 2023. [Online]. Available: https://dl.gi.de/handle/20.500.12116/43045

[43] R. Majumdar, D. Madan, D. Bhoumik, D. Vinayagamurthy, S. Raghunathan *et al.*, “Optimizing ansatz design in qaoa for max-cut,” 2021. [Online]. Available: https://arxiv.org/abs/2106.02812

[44] M. Schönberger, M. Franz, S. Scherzinger, and W. Mauerer, “Peel | pile? cross-framework portability of quantum software,” 2022. [Online]. Available: https://arxiv.org/abs/2203.06289

[45] H. Häffner, C. F. Roos, and R. Blatt, “Quantum computing with trapped ions,” *Physics reports*, vol. 469, no. 4, pp. 155–203, 2008.

[46] D. C. McKay, I. Hincks, E. J. Pritchett, M. Carroll, L. C. G. Govia *et al.*, “Benchmarking quantum processor performance at scale,” 2023.

[47] M. Y. Siraichi, V. F. d. Santos, C. Collange, and F. M. Q. Pereira, “Qubit allocation,” in *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, ser. CGO 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 113–125. [Online]. Available: https://doi.org/10.1145/3168822

[48] K. Yamanaka, E. D. Demaine, T. Ito, J. Kawahara, M. Kiyomi *et al.*, “Swapping labeled tokens on graphs,” *Theoretical Computer Science*, vol. 586, pp. 81–94, 2015, fun with Algorithms. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397515001656

[49] Y. Hirata, M. Nakanishi, S. Yamashita, and Y. Nakashima, “An efficient method to convert arbitrary quantum circuits to ones on a linear nearest neighbor architecture,” in *2009 Third International Conference on Quantum, Nano and Micro Technologies*, 2009, pp. 26–33.

[50] C. Zhang, A. B. Hayes, L. Qiu, Y. Jin, Y. Chen *et al.*, “Time-optimal qubit mapping,” in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 360–374. [Online]. Available: https://doi.org/10.1145/3445814.3446706