

# Path Matters: Industrial Data Meet Quantum Optimization

Lukas Schmidbauer<sup>1</sup>  
Technical University of  
Applied Sciences Regensburg  
Regensburg, Germany  
lukas.schmidbauer@othr.de

Carlos A. Riofrío<sup>2</sup>  
BMW AG  
Munich, Germany  
carlos.riofrío@bmwgroup.com

Florian Heinrich<sup>2</sup>  
BMW AG  
Munich, Germany  
florian.heinrich@bmw.de

Vanessa Junk<sup>1</sup>  
OptWare GmbH  
Regensburg, Germany  
vanessa.junk@optware.de

Ulrich Schwenk<sup>1</sup>  
OptWare GmbH  
Regensburg, Germany  
ulrich.schwenk@optware.de

Thomas Husslein<sup>1</sup>  
OptWare GmbH  
Regensburg, Germany  
thomas.husslein@optware.de

Wolfgang Mauerer<sup>1</sup>  
Technical University of  
Applied Sciences Regensburg  
Siemens AG, Technology  
Regensburg/Munich, Germany  
wolfgang.mauerer@othr.de

**Abstract**—Real-world optimization problems must undergo a series of transformations before becoming solvable on current quantum hardware. Even for a fixed problem, the number of possible transformation paths—from industry-relevant formulations through binary constrained linear programs (BILPs), to quadratic unconstrained binary optimization (QUBO), and finally to a hardware-executable representation—is remarkably large. Each step introduces free parameters, such as Lagrange multipliers, encoding strategies, slack variables, rounding schemes or algorithmic choices—making brute-force exploration of all paths intractable. In this work, we benchmark a representative subset of these transformation paths using a real-world industrial production planning problem with industry data: the optimization of work allocation in a press shop producing vehicle parts. We focus on QUBO reformulations and algorithmic parameters for both quantum annealing (QA) and the Linear Ramp Quantum Approximate Optimization Algorithm (LR-QAOA). Our goal is to identify a reduced set of effective configurations applicable to similar industrial settings. Our results show that QA on D-Wave hardware consistently produces near-optimal solutions, whereas LR-QAOA on IBM quantum devices struggles to reach comparable performance. Hence, the choice of hardware and solver strategy significantly impacts performance. The problem formulation and especially the penalization strategy determine the solution quality. Most importantly, mathematically-defined penalization strategies are equally successful as hand-picked penalty factors, paving the way for automated QUBO formulation. Moreover, we observe a strong correlation between simulated and quantum annealing performance metrics, offering a scalable proxy for predicting QA behavior on larger problem instances.

**Index Terms**—Industrial Production Planning, BILP, QUBO, Annealing, LR-QAOA

## I. INTRODUCTION

Optimization problems are widespread in industrial processes. Great effort is usually employed to lowering costs, boosting efficiency, and enhancing production. Among these processes, production and logistics is of special interest as decisions on supply chains, factory placements, and production allocation are of increasingly more complexity in

an interconnected world. Many of these problems can be cast as combinatorial optimization, which are known to be hard to solve as the number of variables increases. Solving these problems at large scale presents a challenge for current computation paradigms, that is, classical super computers.

Quantum computing promises to help dealing with limits of classical computing. In fact, many algorithms for solving combinatorial problems have been proposed, for instance, the Quantum Approximate Optimization Algorithm (QAOA) [1], recursive QAOA (rQAOA) [2], [3], linear-ramp QAOA (LR-QAOA) [4], adiabatic quantum optimization or quantum annealing [5], and counter adiabatic QAOA (CA-QAOA) [6], which could help alleviating the scaling limitations of classical algorithms. For a review of QAOA and its variants see [7] and for a practical description of problem formulations see [8]. Despite recent progress, it is still unclear whether quantum optimization will be able to solve industry relevant problems. In fact, few classes of problems are known to have guaranteed quantum advantage [9] and others show promising scaling [10]. Most of these algorithms have been implemented in quantum computers for small problem instances with varying levels of success. For example a variant of QAOA has been used to solve instances of the Max-Cut problem up to 127 qubits in quantum hardware [11]. Currently, due to cloud access to quantum computers, researchers are able to routinely test and deploy optimization algorithms in small scale quantum computers, so-called NISQ (noisy intermediate-scale quantum) [12], [13] devices.

Most experiments and demonstrations of quantum algorithms are carried out with a reduced class of problems, for instance, Max-Cut, Max-SAT [14], or traveling salesperson problem (TSP) [15], which are meant to be representative problems (and suitable abstractions) that are hard to solve classically. However, industrial problems seldom fall exactly in a standard category. In this work, we compare the performance of quantum and simulated annealing, as well as the

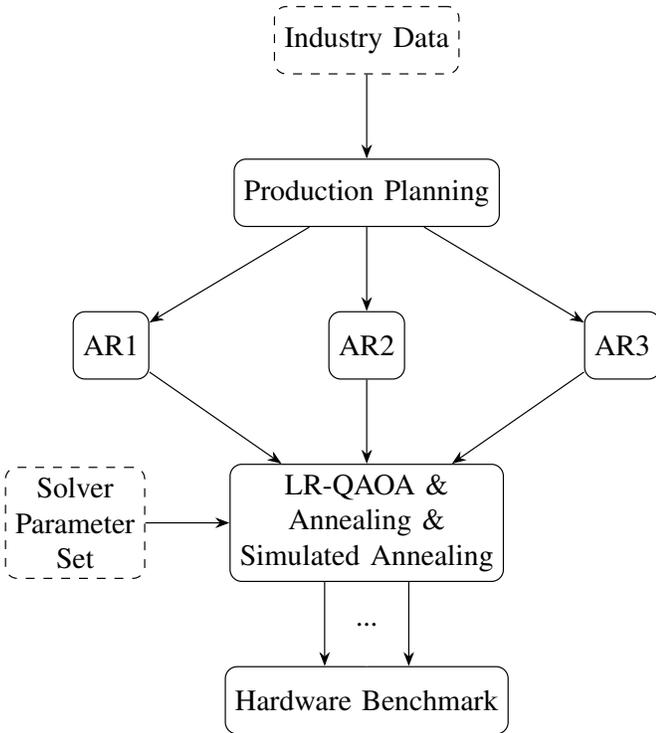


Figure 1: Processing stages overview. Algebraic representations (AR) devised from the Production Planning use case are cast into hardware executable representations for LR-QAOA, (quantum) annealing and simulated annealing.

Linear Ramp Quantum Approximate Optimization Algorithm (LR-QAOA), ran in quantum hardware, when deploying an industry-relevant production and logistics use case: the capacity planning for production of the body parts of a vehicle. Our problem is formulated with real production data and real-world constraints, which in general differ from standard classes of optimization problems. Fig. 1 gives an overview of the general data flow in terms of abstraction layers. While the real industry data forms the input of the parameterized production planning use case, there are a multitude of possibilities to cast that problem into a hardware executable form. We compare different problem encodings (see Fig. 1: AR1, AR2, AR3) and investigate how the quality of solutions varies with them. In particular, we vary the formulation of penalty constraints and data refinement, while also considering different solver specific parameters. We aim to bridge the gap in benchmarking quantum hardware including non-standard, industry-relevant problems.

The remainder of the paper is divided as follows: While Sec. II discusses the current state of the art, Sec. III introduces the necessary algorithmic building blocks. We go into detail about the formulation of the industry relevant use case in Sec. IV. Although we focus on 3 different algebraic representations, experiments have many more free parameters to optimize for cost, solution quality or time-to-solution. Sec. V describes which parameters form the basis for the empirical

evaluation in Sec. VI. Sec. VII concludes our study and suggests future improvements.

This paper is augmented by a comprehensive [reproduction package](#) (link in PDF) that also allows for extending our work. It also provides additional detailed figures about our experiments.

## II. RELATED WORK

Nenno and Caspari [16] analyze the general steps needed for using quantum computers on dynamic optimization problems with respect to an industry-relevant simplified chemical reactor by incorporating a Quadratic Unconstrained Binary Optimization (QUBO) formulation. The QUBO is based on a system of differential-algebraic equations that is embedded into the optimization problem—opposed to former methods that require analytic or parametric solutions [17], [18].

Formulating problems as QUBO or more general as pseudo boolean functions (*i.e.*, polynomials) is not a trivial task. For example, Häner *et al.* [19] outline the challenges for evaluating non-polynomial functions on quantum hardware. Apart from implementation challenges, highly industry relevant join-ordering problems and their performance-critical formulation into QUBO form for database query optimization can be found in [20].

To map such problems into QUBO form, it is necessary to identify and map (in-)equality constraints, discretize continuous variables, apply Lagrange factors, etc.. Glover *et al.* [21] provide an extensive study on transforming optimization problems (*e.g.*, quadratic assignment, knapsack, constraint satisfaction or max-cut) into QUBO form. From the point of view of a real application, mapping to QUBO requires identifying unambiguous similarities to optimization problems. For example, Schütz *et al.* [22] show how to cast robot trajectory planning into QUBO form. Apart from direct QUBO mappings, firstly mapping to Polynomial Unconstrained Binary Optimization (PUBO) and then transforming to QUBO can be a valid choice that, however, impacts non-functional requirements and changes problem specific characteristics. We shed light on this method for a Job-Shop Scheduling problem by using automatic means of transformation [23]. These algebraic representations form the basis for further transformations down to a hardware executable representation (see Fig. 1)—for instance, selecting a solver strategy.

Hauke *et al.* [24] and Yarkoni *et al.* [25] give a broad overview of industrial applications and perspectives using quantum annealing—including traffic flow, scheduling, quantum simulation and finance. A recent work by Vandelli *et al.* [26] simulates QAOA and annealing for power consumption in telecommunication networks for up to 31 qubits. They show that finding near-optimal solutions is possible—even for constrained problems with 31 qubits. Krol *et al.* [27] present a quantum version of an industrial shift scheduling problem via another solving strategy, that is, Grover’s search.

### III. FUNDAMENTALS

#### A. LR-QAOA

QAOA, originally proposed by Farhi *et al.* [1], is a hybrid quantum classical algorithm. Apart from using QAOA as a solver for Optimization Problems (OPs), Morales *et al.* [28] consider the conditions for universal computations. The quantum circuit for QAOA consists of  $p$  layers of a problem specific part  $H_C(\gamma_i)$  and a mixer  $H_M(\beta_i)$ , parametrized by rotation angles  $\gamma_i$  and  $\beta_i$ . These angles are subject to a classical optimizer, which requires multiple executions of the quantum circuit to find a (local) minimum of an optimization problem, encoded in  $H_C$ .

In contrast to QAOA, LR-QAOA [29] uses predetermined values for rotation angles in each layer. The values for  $\gamma_i$  (the problem specific part) increase linearly, while the values for  $\beta_i$  (the mixer) decrease linearly with each layer, respectively. However, this necessitates normalizing the objective function, which is computationally feasible in polynomial time. Since quantum annealers can also realize linear schedules, comparing them to LR-QAOA is interesting in view of industrial use cases in terms of performance, solution quality, resource requirements, scaling behavior, as well as characterizing the impact of problem formulations on these measures.

#### B. Adiabatic Quantum Computing

Contrary to discretized circuit-based models of quantum computation, adiabatic quantum computing is a continuous process. Nevertheless, both models of computation are polynomially equivalent in their computational power [30]. When slowly evolving a time-dependent Hamiltonian  $H(\tau)$  ( $\tau \in [0, T]$ ), while starting in the ground state at  $\tau = 0$ , the ground state is preserved at  $\tau = T$  with probability  $P$ . The probability  $P$  is close to 1 when the spectral gap  $\delta(\tau)$  (*i.e.*, the absolute eigenvalue difference between the first excited state and the ground state<sup>1</sup>) is strictly greater than 0 ( $\forall \tau \in [0, T]$ ) and the process evolves slowly [31]: Let  $n$  denote the problem size and let  $\delta_m$  denote the minimum spectral gap. Then, the time  $T$  is polynomial in  $n$ , iff  $\delta_m$  is inverse polynomial in  $n$ .

The time-dependent Hamiltonian

$$H(\tau) = \frac{T - \tau}{T} H_{\text{init}} + \frac{\tau}{T} H_{\text{final}} \quad (1)$$

can be split into an easy to prepare initial Hamiltonian  $H_{\text{init}}$  (with known ground state) and a Hamiltonian  $H_{\text{final}}$  that encodes the optimization problem. While Eq. 1 interpolates linearly between  $H_{\text{init}}$  and  $H_{\text{final}}$ , other annealing schedules are possible. Unfortunately, finding the minimum spectral gap to adjust the schedule is a hard problem with recent development in terms of mitigating anti-crossings [32]–[34].

The D-Wave Quantum Annealer is a non-universal quantum computer in the sense that not every possible state in the

Hilbert Space can be reached. This is a result of the time-dependent Hamiltonian that is realized in hardware [35]:

$$H = -\frac{A(\tau)}{2} \sum_i \sigma_x^{(i)} \quad (2)$$

$$+ \frac{B(\tau)}{2} \left[ \sum_i h_i \sigma_z^{(i)} + \sum_{i < j} J_{ij} \sigma_z^{(i)} \sigma_z^{(j)} \right], \quad (3)$$

where  $\sigma_x^{(i)}$  and  $\sigma_z^{(i)}$  denote the Pauli-X and -Z operations applied to qubit  $i$ , respectively. Similarly to Eq. 1,  $A(\tau)$  and  $B(\tau)$  adjust the influence of the initial Hamiltonian (Eq. 2) and the problem Hamiltonian (Eq. 3). In particular, we can choose  $A(\tau)$  and  $B(\tau)$  to represent an approximately linear annealing schedule. Although the D-Wave Quantum Annealer cannot reach any possible state in Hilbert Space, it is a universal model of computation in the sense of solving NP-complete problems due to solving QUBOs. As a side note, by exploiting level crossings, one can implement gate operations beyond  $\sigma_z$  and  $\sigma_x$  [36].

#### C. Simulated Annealing

Simulated annealing is a probabilistic classical method to solve optimization problems [37] and in particular Pseudo-Boolean Functions (PBFs). Therefore, this method can serve as a classical baseline for experiments. In essence, one chooses a random initial state and then repeats the following  $n$  times: Firstly, choose a random variable in the current bit string and flip it<sup>2</sup>. If this flip lowers the energy of the PBF<sup>3</sup>  $f$ , accept that flip. Conversely, if this flip increases the energy of  $f$ , accept this flip based on a random experiment. The probability to accept the flip depends on the energy difference as well as a typically decreasing base probability [31]. For example, one can choose the probability  $P$  as  $P = \min(1, e^{-(\Delta/i)})$ , where  $\Delta$  denotes the energy difference to the function without the bit flip and  $i$  is the iteration count. If the random experiment does not accept the flip, revert it.

### IV. USE CASE MODELLING

#### A. Description and Mathematical Formulation

In the complex process of vehicle mass production, forming of raw materials into product parts is an early and fundamental step. One of such processes is done in the *press shop*, in which metal sheets are shaped into the components of the body of a vehicle. Press shops may house several *press machines*, which provide the force by which the metal is reshaped and cut using *toolkits*, which are the tooling components with the shape of body parts. Press shops are distributed globally and do not necessarily coincide with the assembly plants, which put together the whole vehicle. Production and logistic costs are associated with every choice of production allocation as well as legal and technical constraints regarding volume of parts which can be produced at a given time and location. For example, each press machine has a maximum capacity it can

<sup>1</sup>We assume a non-degenerate ground state of  $H(\tau)$ .

<sup>2</sup>Deterministically choosing the next variable is also possible.

<sup>3</sup>For minimization problems.

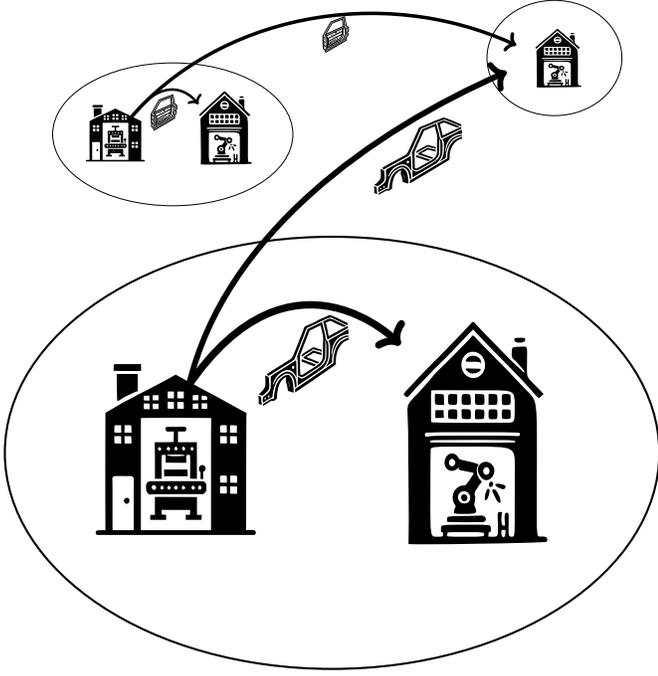


Figure 2: An exemplary scheme of three production sites, two comprising a press shop and an assembly plant, one with an assembly plant only, sized to convey an impression of distance (smaller means farther away). Toolkits are assigned to the press machines of the press shops, thus determining where which parts are pressed and where consequently the flows of produced parts to the assembly plants must originate from. Product flows of a door and a side-frame can be seen, each fulfilling onsite as well as offsite demands.

offer during a given production period and each toolkit must produce a given number of parts. Figure 2 gives an illustration of the setting.

The question for long-term planning is to decide which toolkits should be assigned to which press machines in a way that is cost optimal. The decisions, for example, how many parts to produce or which toolkit to use for the production of which part, are given parameters and not degrees of freedom in finding optimal solutions. Therefore, each toolkit can be assumed to “carry” a given amount of parts demanded, which, via press-machine dependence and given work rates, translates into an effective (possibly press-machine dependent) workload. Thus, the allocation decision can be cast as an optimization problem in which the total production costs are minimized. The minimization is subject to all parts being produced while respecting the maximum capacities of every machine.

This optimization problem can be formulated as a binary constrained linear program. We define  $T$  as the set of all toolkits and  $M$  as the set of all machines. Then, we can introduce the decision variable  $x_{tm}$  to encode which toolkit

$t \in T$  is assigned to which machine  $m \in M$ ,

$$x_{tm} = \begin{cases} 1 & \text{if toolkit } t \text{ is assigned to machine } m, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

The cost arising by an assignment is given by  $c_{tm}$ , leading to the objective function

$$\min_{\mathbf{x}} \sum_{t \in T} \sum_{m \in M} c_{tm} x_{tm}, \quad (5)$$

where  $\mathbf{x} = (x_{tm})_{t \in T, m \in M} \in \{0, 1\}^{T \times M}$ . Each machine has a maximum capacity  $h_m$ , usually given in units of hours per production period, and each toolkit assignment requires a demanded workload  $w_{tm}$ , also in hours per production period, resulting in the capacity constraints

$$\sum_{t \in T} w_{tm} x_{tm} \leq h_m, \quad \forall m \in M. \quad (6)$$

Moreover, each toolkit has to be assigned to a machine exactly once, which is encoded as

$$\sum_{m \in M} x_{tm} = 1, \quad \forall t \in T. \quad (7)$$

This constraint ensures that each toolkit is assigned to exactly one location and also prevents that nothing is produced despite of existing demands. Note that producing nothing would be the cheapest option concerning the objective Eq. 5.

We use this linear program both for deriving the problem formulation for the quantum computer and for computing the optimal solutions of the problem instances in Sec. V. For the latter, we use the SCIP – short for Solving Constraint Integer Programs – solver [38] via the pywraplp interface from Google OR-Tools [39]. This is done for comparison purposes.

## B. QUBO Creation and Penalty Terms

The standard way to run an optimization problem in a quantum computer is to reformulate it as a QUBO. The procedure of transforming the original, constrained, optimization (Eq. 5, 6 and 7) into an unconstrained problem is to add the constraint violations as penalty terms to the objective function (Eq. 5). For our reformulation, we use the Qiskit optimization library (version 0.6.0) [40]. In the following, we sketch the resulting steps.

First, inequality constraints, in our case the capacity-constraints (Eq. 6), are transformed into equality constraints. The transformation is achieved by introducing so-called “slack variables”, which we denote by  $S_m$  since there is one capacity constraint for each press machine. The reformulated capacity constraints read

$$\sum_{t \in T} w_{tm} x_{tm} + S_m - h_m = 0, \quad \forall m \in M, \quad (8)$$

where  $0 \leq S_m \leq h_m$ , since the lower bound of  $\sum_{t \in T} w_{tm} x_{tm}$  is zero. Next, the newly introduced slack

variables  $S_m$  have to be split into a binary representation to fulfill the requirements of a QUBO,

$$S_m = \sum_{j=0}^{r_m-1} 2^j s_{mj} + (h_m - 2^{r_m} + 1) s_{mr_m}, \quad (9)$$

extending the problem by the binary variables  $s_{m0}, \dots, s_{mr_m}$  per press machine  $m$ , where  $r_m = \lfloor \log_2(h_m) \rfloor$ .

Now, all constraints are equalities and can be transformed into penalties of parabolic shape that are added to the objective function. For the assignment constraints (Eq. 7) we obtain the penalty terms

$$\lambda_t \left( \sum_{m \in M} x_{tm} - 1 \right)^2, \quad \forall t \in T, \quad (10)$$

with penalty factors  $\lambda_t$  and for the capacity constraints (Eq. 6) we write the penalty terms

$$\lambda_m \left( \sum_{t \in T} w_{tm} x_{tm} + S_m - h_m \right)^2, \quad \forall m \in M, \quad (11)$$

with penalty factors  $\lambda_m$  and  $S_m$  substituted by Eq. 9. The total QUBO problem is then given by the objective function

$$\min_{\mathbf{x}, (s_m)_{m \in M}} \left[ \sum_{t \in T} \sum_{m \in M} c_{tm} x_{tm} + \sum_{t \in T} \lambda_t \left( \sum_{m \in M} x_{tm} - 1 \right)^2 + \sum_{m \in M} \lambda_m \left( \sum_{t \in T} w_{tm} x_{tm} + S_m - h_m \right)^2 \right], \quad (12)$$

with  $S_m$  given by the binary variables  $s_{mj}$  defined in Eq. 9, combined to the binary vector  $\mathbf{s}_m = (s_{m0}, \dots, s_{mr_m})$ . The QUBO problem (Eq. 12) can also be brought into matrix form,

$$\min_{\mathbf{x} \in \{0,1\}^N} \mathbf{x}^T \hat{\mathbf{Q}} \mathbf{x} + C, \quad (13)$$

where the binary vector  $\mathbf{x}$  collects all decision variables  $x_{tm}$  and slack variables  $s_{mj}$ , and the constant  $C$  subsumes all constants. Since  $x^2 = x$  for binary variables  $x$ , all coefficients of the terms of degree 1 and 2 in the variables  $x_{tm}$  and  $s_{mj}$  can be collected in the QUBO matrix  $\hat{\mathbf{Q}}$ .

So far, we have not discussed how to choose the penalty factors  $\lambda_t$  and  $\lambda_m$ . Moreover, there are different possibilities for encoding the data of actual problem instances into the linear program and thus the QUBO. In Sec. V we present three different approaches for building the QUBO which we benchmark on quantum hardware in Sec. VI.

## V. EXPERIMENTAL SETUP

### A. Problem instances

For our experiments, we create 6 problem instances from anonymized production data. These problem instances are small enough to fit in a quantum computer, but the values of costs and capacities are taken to represent real situations. That means that the costs, capacities, and restrictions are quantitatively comparable with real production data. The only

adjustment we make is to safely round the values entering the capacity constraint to integers, that is, the maximum machine capacity  $h_m$  is rounded down and the capacity  $w_{tm}$  required by a toolkit assignment is rounded up. We summarize the problem instances in Tab. I.

Table I: Problem instances.

Toolkits	Machines	Qubits
3	2	22
9	2	36
13	2	46
16	2	54
18	2	58
19	2	60

All problem instances are represented by 3 different QUBO encodings (see Fig. 1: AR1, AR2, AR3):

1) *Raw QUBOs*: We take the data directly without any processing and create QUBO formulations for each problem instance. We do not optimize the penalty factors. Instead, we choose them in a broad grid  $\lambda_m \in \{10^3, 10^4, 10^5\}$  and  $\lambda_t \in \{10^7, 10^8, 10^9\}$ , taking the same values for all machines  $m$  and toolkits  $t$ , respectively.

2) *Scaled QUBOs*: We adjust the scaling of the assignment constraint by multiplying Eq. 7 with  $\lambda_s \in \{0.1, 1\}$ . Moreover, before transforming the constraints to penalties but after transforming inequalities to equalities, we rescale the constraints from Eq. 7 and Eq. 8 such that they obtain a similar value range as the objective function in Eq. 5. We define the value range  $v$  of a term  $f$  as

$$v = |[f] - \lfloor f \rfloor|. \quad (14)$$

For the rescaling, we first determine the largest value range  $v_{\max}$  occurring in the linear program by evaluating Eq. 14 for the objective function in Eq. 5 and the left-hand side of the constraints in Eq. 8 and Eq. 7 (after multiplying Eq. 7 with  $\lambda_s$ ). Then, we divide both the objective function, Eq. 5, and all constraints, Eq. 7 and Eq. 8, by their respective range  $v$  and then multiply them by  $v_{\max}$ . Afterwards, we proceed with the transformation of the linear program to a QUBO by transforming the constraints to penalties. The rescaling of the constraints replaces optimizing the penalty factors, such that  $\lambda_m = \lambda_t = 1$  for all  $m \in M$  and  $t \in T$ .

3) *Rounded-cost QUBOs*: For this formulation, we first take the data and rescale the production costs so that the minimum cost is 1. To avoid non-integer cost values, we employ integer division for this rescaling. Then, we proceed with the same transformation routine as described for the *scaled QUBOs*. The aim of rescaling the costs is that the resulting QUBO matrix should be more balanced than for the first two formulations—meaning that the difference between the minimum and maximum value in the QUBO matrix is significantly reduced.

### B. Methods

Evaluating multiple problem instances with multiple solver strategies and (quantum) hardware represents paths in the

abstraction layer graph (compare to Fig. 1). On top of choosing a path, free parameters in transformations influence properties and ultimately performance. We always test all mentioned problem sizes (see Tab. I)—independently of the used solver strategy and hardware. Additionally, depending on the used problem instance (*raw*, *scaled* and *rounded*), we test their respective penalty factors for all solver strategies.

For the solver strategy LR-QAOA, we identify the number of layers  $p$  and the variation in the slope defining parameters  $\Delta_\gamma$ ,  $\Delta_\beta$  (see Sec. III) as important parameters. Preliminary tests on hardware indicated that any  $p > 10$  does not lead to improved performance. Hence, we select  $p \in \{1, 2, 5, 10\}$ . Montanez-Barrera and Michielsen [29] test  $\Delta_\gamma$ ,  $\Delta_\beta$ -dependent performance for six optimization problems. Based on their results for pure optimization problems, we choose  $\Delta_\gamma = 0.9$  and  $\Delta_\beta = 0.6$ . Any logical quantum circuit can be depth-optimized in polynomial time—only differing from the optimal solution by at most one [41]. The edge coloring problem and Vizing’s theorem [42] provide the basis for this argument. However, optimizing circuit depth becomes NP-hard, when considering restricted arbitrary hardware topologies. For specific topologies there have been advances [43], in particular for the heavy-hex topology [44]. We employ an almost depth optimal logical circuit (via edge coloring), which is then transpiled to the hardware gate set and topology of `ibm_marrakesh`. Qiskit [45] provides four levels of optimization of which we use level 0 (no optimization) and 3 (highest optimization).

For the solver strategy of quantum annealing, recall that annealing-based approaches are influenced by the minimum spectral gap (see Sec. III). Hence, the annealing time is an important parameter to avoid level crossings. Therefore, we choose the annealing time  $\tau \in \{10, 20, 40, 80, 160, 320, 640, 1280\}[\mu s]$  for the Advantage\_system4.1 QPU that covers 5760 qubits via the Pegasus topology [46]. Note that the maximum allowed annealing time also depends on the number of shots, which we choose to be 500 to improve measurement statistics. Additionally, a custom annealing schedule can accelerate annealing where the spectral gap is large and decelerate annealing where the spectral gap is small. Since solving for an optimal annealing schedule is at least as hard as solving the optimization problem at hand<sup>4</sup>, we use 4 annealing schedules:

- 1) *Linear*: ↘
- 2) *Bowover*: ↘
- 3) *Bowunder*: ↘
- 4) *SteepFlatSteep*: ↘

Picture a linearly decreasing schedule  $s_{Linear}(\tau)$ <sup>5</sup>. Then, *Bowover* represents a schedule above  $s_{Linear}(\tau)$ —leading to slow annealing at the start and faster annealing when progressing. Analogously, *Bowunder* represents a schedule below  $s_{Linear}(\tau)$ —having the exact opposite effect. Finally, *SteepFlatSteep* is a combination of *Bowunder* at the start and *Bowover*

at the end—leading to slower annealing in the middle of the process. To map a given problem instance onto D-Wave’s hardware, we use the heuristic MinorMinor embedding [47]: It finds a minor-embedding of the graph representation of a given QUBO instance in the graph that represents D-Wave’s hardware (see [48] for more information on graph minors). Note that an embedding for a QUBO of size  $|Q|$  usually requires more than  $|Q|$  qubits in D-Wave’s hardware.

Similar to the time parameter in quantum annealing, simulated annealing can employ longer classical runtime (*i.e.*, steps  $n$ ; see Sec. III) to heuristically refine the current local minimum. Also, the probability function can be changed to, for instance, overcome local minima. However, we use simulated annealing as a baseline and thus use the standard geometric schedule and set  $n = 1280$ .

To accommodate for a single bit-flip error, we use a classical post-processing error mitigation technique for all experiments (see [29]): For each bit  $x$  in a sampled bitstring  $\mathbf{x}$ , we test if the negation of  $x$  improves the energy with respect to the QUBO (see Sec. V). We then use  $\mathbf{x}'$  with lowest energy. Note that its runtime scales linearly in the size of the bitstrings and the number of samples.

### C. Expected Behavior

It is evident that current Noisy Intermediate Scale Quantum (NISQ)-era hardware still is restricted by noise. Although Quantum Error Correcting Codes (QECCs) are in development and have provable advantages, their use is to the detriment of requiring more qubits (see [49]). QECCs offer potential to allow for some degree of noise, while protecting the intended quantum state. Hence, (depending on the number of additionally used qubits), QECCs will eventually make noisy quantum hardware behave similar to noiseless systems and noiseless simulations. For the following experimental analysis, we expect bigger problem sizes to perform worse, due to either exponentially scaling search spaces or noise that arises from inherently longer circuits (LR-QAOA) and bigger embedding size (annealing).

As the number of layers  $p$  in LR-QAOA increases, the logical circuit depth increases linearly with  $p$ . While noiseless LR-QAOA converges to an optimal solution (under the assumptions of the adiabatic theorem; see Sec. III), hardware noise eventually leads to a (non-uniform<sup>6</sup>) random output distribution. We confirm the convergence to optimal solutions numerically by simulating the 3 Toolkit (22 qubit) case for up to  $p = 100$  LR-QAOA layers. Notably, even for  $p = 1$  and 1000 shots, we find the optimal solution regardless of the problem variant (*raw*, *scaled* or *rounded*). However, we notice that the *rounded* variant performs better for  $p \geq 5$  than the *raw* and *scaled* variant. The [reproduction package](#) (link in PDF) contains detailed figures. Take into consideration that system noise eventually diminishes better Trotterization accuracy (*i.e.*, higher  $p$ ). Noise can also blur variant-specific effects. Therefore, the free parameter  $p$  is subject to balancing both noise

<sup>4</sup>We would need to know the spectral gap for every  $\tau$ .

<sup>5</sup>A decreasing schedule refers to  $H_{init}$ .  $H_{final}$  is scaled accordingly.

<sup>6</sup>This is hardware dependent (*e.g.*, via topology).

and approximating annealing through Trotterization. Hence, a (variant-dependent) tipping point should be evident, when increasing  $p$ . A similar effect can be expected for the annealing time and the annealing schedule to avoid level-crossings in the adiabatic evolution. They are also free parameters that need to balance the expected noiseless performance gain and the effects of noise.

Concerning the different QUBO formulations, *raw*, *rounded*, and *scaled*, the number of effective hardware runs differs for each of the variants due to the QUBO’s respective penalization strategy. For *raw* we have 9 combinations; for *scaled* 2 combinations, and for *rounded* 1 combination. If our experiments were mere random sampling and all effects of the quantum circuit on the final state measurement were washed out by noise, we would expect the *raw* QUBO to perform best simply due to being run more often than the other versions and thus having the highest probability of sampling a good result eventually. Of course, we expect the quantum circuit to have a measurable effect on the final quantum state in our experiments and the different penalization strategies alter the QUBO formulation and should hence perform differently. Nevertheless, the different number of runs could result in a (slight) bias in favor of the *raw* variant.

## VI. EXPERIMENTAL RESULTS

For each of the following combined figures, we show results for quantum annealing on top and results for LR-QAOA below. Each figure has the problem size (*i.e.*, the number of toolkits) on its (non-equally-spaced) x-axis. Note that toolkits directly translate to number of qubits, since we always use 2 press machines (see Tab. I). Data points in each figure must adhere to its respective criteria (*e.g.*, being valid). Consequently, if there are no data points for toolkits  $t$ , we omit  $t$  in the figure.

For Fig. 3, 4 and 5, we pick the best performing penalty for each annealing time or layer  $p$  (see Sec. V-B) per toolkit and problem formulation and then show their values in a boxplot. Its box is bounded by the first and third quartile and additionally shows the median as a line. Furthermore, its whiskers extend to at maximum  $1.5 \cdot \text{IQR}$  (depending on actually available data points), where the interquartile range (IQR) is the distance between the first and third quartile. Any data point outside this range is shown as a (partially translucent) circle ( $\bullet$ ). Moreover, we show the best performing penalty for each x-value (*i.e.*, toolkits) explicitly as a shape: For the *raw* variant, a (rotated) rectangle ( $\square$ ,  $\diamond$ ) corresponds to  $\lambda_m = 3$  and a rotated triangle ( $\triangle$ ,  $\nabla$ ) corresponds to  $\lambda_m = 4$ . Their respective rotation encodes  $\lambda_t$ . The *scaled* variant only has two penalties, which we show as a circle with a cross ( $\lambda_s = 0.1$ ) or a circle with an x ( $\lambda_s = 1$ ). Analogously, we show the *rounded* variant as a circle. Small problem sizes lead to many best performing penalty weights and hence we omit explicit shapes for 3 and 9 toolkits. For a comparison to classical methods, we use simulated annealing (1280 steps) and the best out of 1000 randomly generated bitstrings as reference. Although we test four annealing schedules, as described in Sec. V, we only show the linear schedule due to its similarity

to LR-QAOA. Note that the type of our test schedules only has minor influence on the performance. Similarly, for LR-QAOA we test optimization level 0 and 3 in Qiskit, but restrict Fig. 3, 4, and 5 to level 3. Take into consideration that optimization level 0 usually worsens the results—with outliers that probably originate from system noise.

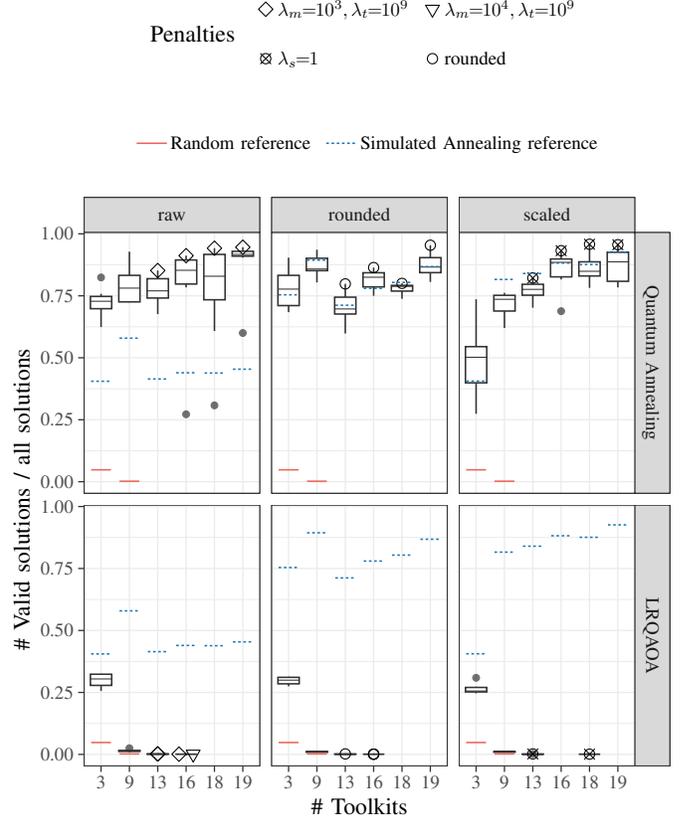


Figure 3: Problem size (x-axis) vs percentage of valid (*i.e.*, constraint satisfying; see Sec. IV) solutions (y-axis) and corresponding best performing penalties (see Sec. V). Closer to 1.00 is better. Horizontal facets: quantum annealing on D-Wave and LR-QAOA on IBM. Vertical facets: problem formulation (see Sec. IV and Fig. 1). Random (solid line) and Simulated Annealing (dotted line) as reference (same values for both horizontal facets).

Fig. 3 shows the percentage of valid solutions (closer to 1.00 is better). Valid solutions satisfy the problem constraints given in Sec. IV (*i.e.*, capacity constraint, Eq. 6, and assignment constraint, Eq. 7). For both LR-QAOA and quantum annealing and each QUBO variant (*raw*, *rounded*, and *scaled*), we see that a specific set of penalties leads to the highest percentage of valid solutions for bigger problem instances. Note that we test 9 penalty combinations for the *raw*, two for the *scaled*, and one for the *rounded* variant, due to their different construction. LR-QAOA rarely finds valid solutions for bigger problem instances. Contrary, annealing performs exceptionally well—finding  $\geq 90\%$  valid solutions for bigger problem instances. Interestingly, the simulated annealing reference is Pearson-

correlated [50] to the mean of the quantum annealing results:  $r_{raw} \approx 0.2736$ ,  $r_{rounded} \approx 0.9297$  and  $r_{scaled} \approx 0.9659$ . Hence, the *rounded* and *scaled* variant, seem to have beneficial numerical nature for simulated and quantum annealing, which allows for some degree of extrapolation.

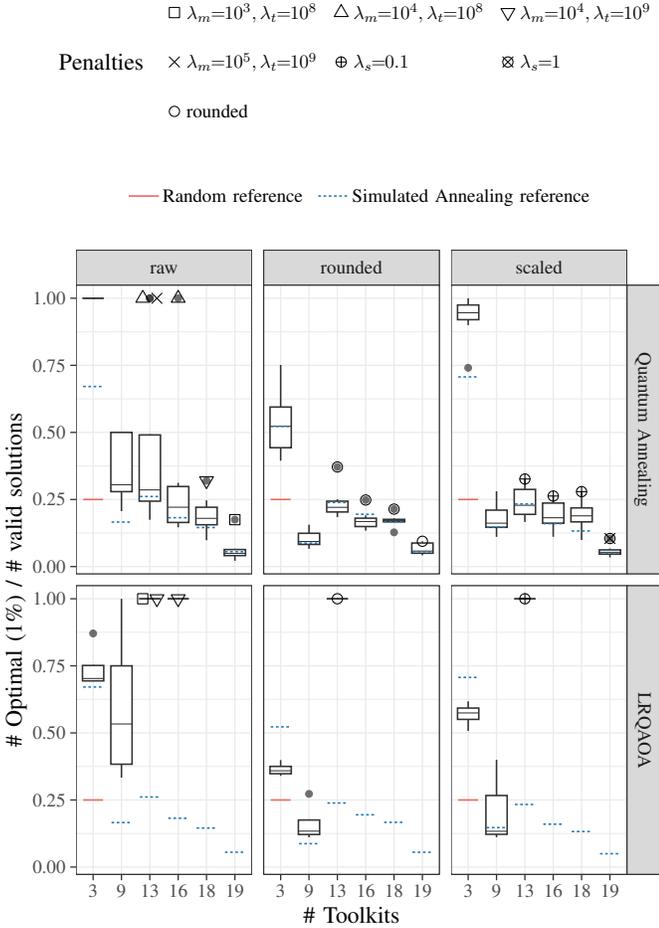


Figure 4: Problem size (x-axis) vs ratio of valid (*i.e.*, constraint satisfying; see Sec. IV) solutions within 1% of the optimum to valid solutions (y-axis) and corresponding best performing penalties (see Sec. V). Closer to 1.00 is better. Horizontal facets: quantum annealing on D-Wave and LR-QAOA on IBM. Vertical facets: problem formulation (see Sec. IV and Fig. 1). **Random** (solid line) and **Simulated Annealing** (dotted line) as reference (same values for both horizontal facets).

Although Fig. 3 reduces the set of solutions to valid solutions, they can be numerically far from optimal in terms of production cost. Hence, Fig. 4 restricts the set of valid solutions to solutions that lie within 1% of the optimal solution (closer to 1.00 is better). A higher share of (near) optimal solutions also leads to faster time-to-solution, which is cost beneficial for computation. Smaller problem instances have a higher share of optimal solutions—potentially due to the exponentially scaling solution space, see random reference (red) that only finds valid solutions for 3 and 9 toolkits and only close-to-optimal solutions for 3 toolkits. LR-QAOA

fails to find (near) optimal solutions for 18 and 19 toolkits, while quantum annealing finds (near) optimal solutions for all tested toolkits—although their share decreases with increasing problem size. As before, we calculate the Pearson correlation [50] between simulated annealing and the average quantum annealing results:  $r_{raw} \approx 0.9874$ ,  $r_{rounded} \approx 0.9965$  and  $r_{scaled} \approx 0.9960$ . Note that Fig. 4 does not show absolute values and depicts a subset of Fig. 3. By multiplying the percentage of valid solutions (Fig. 3) by the share of (near) optimal to valid solutions (Fig. 4), one can obtain the percentage of (near) optimal solutions, which further highlights the difference between quantum annealing on D-Wave and LR-QAOA on current IBM devices.

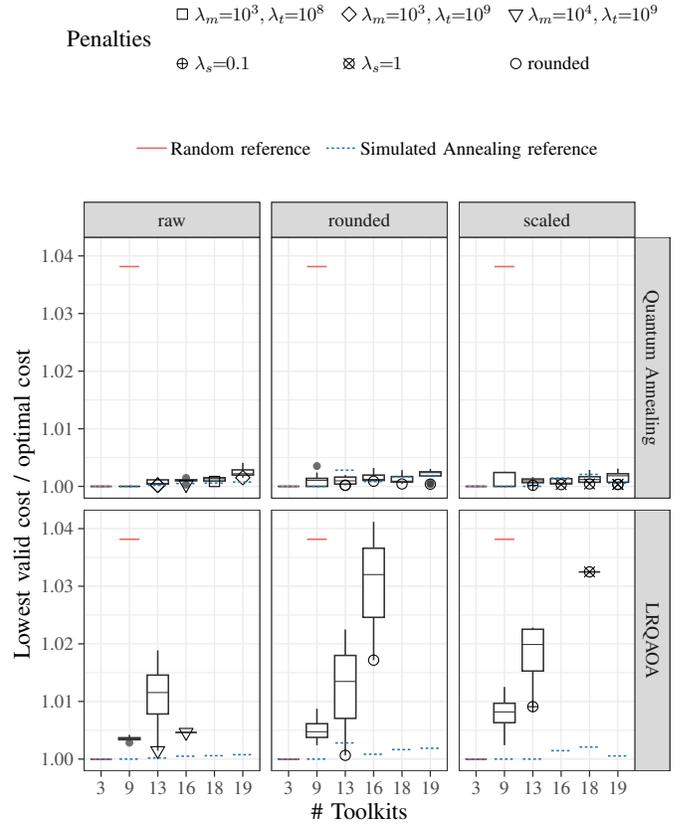


Figure 5: Problem size (x-axis) vs normalized lowest cost for valid (*i.e.*, constraint satisfying; see Sec. IV) solutions (y-axis) and corresponding best performing penalties (see Sec. V). Closer to 1.00 is better. Horizontal facets: quantum annealing on D-Wave and LR-QAOA on IBM. Vertical facets: problem formulation (see Sec. IV and Fig. 1). **Random** (solid line) and **Simulated Annealing** (dotted line) as reference (same values for both horizontal facets).

Fig. 3 and 4 give insights in the distribution of solutions and therefore allow to extrapolate the time-to-solution or runs required to obtain a sufficiently good solution. Ultimately, the solution with the least cost is of interest for production. Therefore, Fig. 5 gives the ratio for the lowest valid to the optimal cost (closer to 1.00 is better). Although the results

for quantum annealing all lie within 0.41% of the optimal solution, the difference in problem formulation is apparent. At 19 toolkits, the *raw* variant performs worse than the *rounded* and *scaled* variants. For completeness, we calculate the Person correlation [50] between the mean value for annealing and simulated annealing [50]:  $r_{raw} \approx 0.9516$ ,  $r_{rounded} \approx 0.4224$  and  $r_{scaled} \approx 0.4184$ . As LR-QAOA produces little to none valid solutions for increasing problem sizes (see Fig. 3), they are increasingly farther from the optimal solution. Nevertheless, the final measured state obtained from LR-QAOA still contains problem specific information, since LR-QAOA is dramatically better than the random reference, which cannot find valid solutions for more than 9 toolkits. Hence, system noise does not render the final state useless. Upcoming implementations of error correction should therefore make a significant contribution to improving the results. Although, a higher annealing time has a slight net positive effect on the QUBO energy, with outliers potentially originating from system noise, we do not find a strong influence of annealing time on QUBO energy. Hence, the spectral gap does not appear to be the limiting factor for the tested problem sizes. This also applies to LR-QAOA analogously.

Overall, quantum annealing outperforms LR-QAOA for the tested problem instances. However, scaling behavior beyond the capabilities of current hardware can put that result into a different perspective. For LR-QAOA, we identify the number of transpiled non-local gates (*i.e.*, two qubit gates in our case) as the relevant metric. Note that the circuit depth, the number of total gates and the number of used qubits can also be valid metrics. When considering the effect of transpilation, missing connections in hardware are resolved by introducing additional (non-local) gates. Hence, a gate-based approach tends towards deeper circuits, while the number of qubits is not a hard limiting factor: Given a quantum hardware with  $q_h$  many qubits that form a connected graph. Then, for any given logical quantum circuit with  $q_l \leq q_h$  qubits, there exists a unitary-equivalent (deep) hardware executable circuit (see [51] or [52]). Contrary, for quantum annealing, we use the embedding size on hardware as the eventually limiting factor, when increasing problem size, since missing connections are resolved by combining hardware qubits [53]. Fig. 6 shows these metrics on the y-axis. The x-axis shows the number of toolkits (we omit the label for 18 toolkits, but show its data). It is linearly spaced by the number of qubits, since they correspond to toolkits (see Tab. I). In LR-QAOA, the number of layers  $p$  increases the number of non-local gates in the logical circuit linearly. We found that this is similar for the transpiled circuit—meaning that the transpiler has no significant impact in combining layers. Hence, we only show  $p = 1$  for optimization levels 0 and 3. Optimization level 3 roughly halves the number of transpiled gates and therefore has significant impact on solution quality that is highly influenced by gate noise. Compared to LR-QAOA, quantum annealing has a higher spread in embedding size, with the least spread for the *rounded* variant—albeit there being only minor differences between problem formulations.

Take into consideration that the x-axis is spaced in terms of qubits, but labeled with toolkits (see Tab. I), which allows for comparing the scaling behavior with other industry relevant problems. Also take into consideration, that the QPU access time is significantly higher with IBM devices and LR-QAOA, which is relevant for a cost-to-solution estimation. On average, for a single experiment (500 shots), the D-Wave annealer uses  $\approx 0.3$  seconds, while IBM (1000 shots) uses  $\approx 7.8$  seconds.

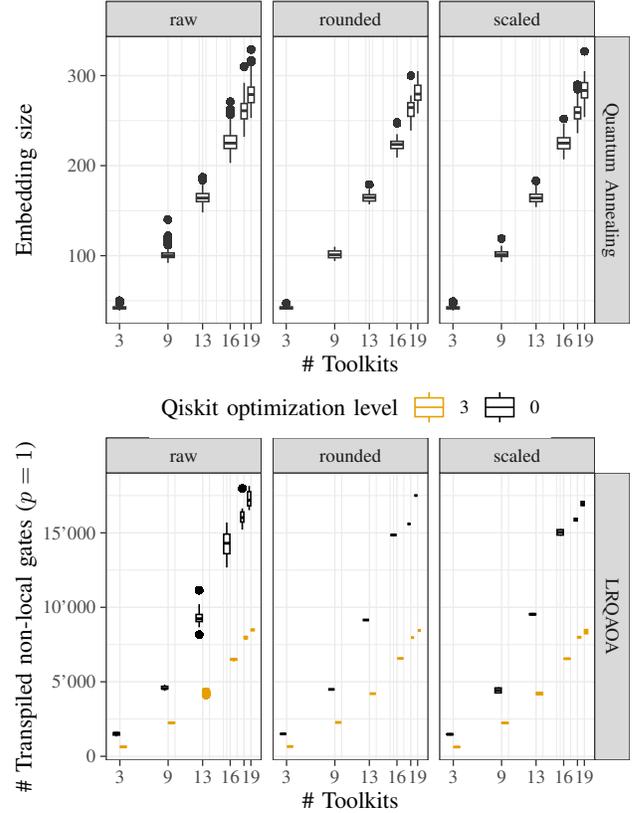


Figure 6: Problem size (x-axis; linearly spaced by # qubits; see Tab. I) vs scaling behavior (*i.e.*, embedding size or number of transpiled non-local gates for  $p = 1$ ; y-axis). Lower is better. Horizontal facets: quantum annealing on D-Wave and LR-QAOA on IBM. Vertical facets: problem formulation (see Sec. IV and Fig. 1). Qiskit optimization level only applies to LR-QAOA.

In summary, the experiments suggest that a choice of penalty weights that clearly outperforms the others emerges with increasing problem size. Hence, for similarly structured problems, it can be beneficial to first find a set of suitable penalty weights by analyzing smaller but growing problem instances. These optimized weights should then also perform well for larger problem instances. Moreover, we find similar or better performance for the *rounded* and *scaled* variant over the *raw* variant. From an industry point of view, it is therefore not necessary to perform a grid search over the basic formulation, but rather automatically select suitable parameters (as in the *scaled* and *rounded* case). On top of that, we show a

high correlation of the average annealing result and simulated annealing for the *scaled* and *rounded* variant. Therefore, it is possible to (locally) extrapolate to higher problem instances by using classical simulated annealing and then use quantum annealing to improve the results.

## VII. CONCLUSION AND OUTLOOK

Creating good QUBO formulations for industry use cases is not a trivial problem: encodings, slack variables, and penalty terms play a crucial role in performance. Formulations that limit the number of qubits, if they exist, should be prioritized. However, in the long term, quantum computers may have enough qubits and be sufficiently robust for naive formulations to work. In this work, we have explored the formulation and solution of an industrial optimization application using quantum computing techniques. We used real anonymized data to describe the problem of allocating production capacity for a network of press shops.

We observed that quantum annealing via D-Wave hardware performs surprisingly well, even for modest-size problem instances. This approach offers great potential, but the embedding dimension scaling with the problem size is not yet ideal. Hopefully, better architectures with better connectivity will become available in the future.

Even modest-size problems require significant quantum resources to work for all algorithms and platforms we tested. Noisy quantum hardware and incomplete connectivity seem to be the main limiting factors for scaling quantum computers to industry-level needs. Compared to QAOA, LR-QAOA is simpler and cheaper to implement—using orders of magnitude less quantum resources in time. However, when deployed on IBM machines, it lags behind the classical baseline of simulated annealing—mainly due to noise. Although the noisy nature of quantum gates is evident from the obtained solutions, they still contain problem-specific information. Moreover, noiseless simulations return good results for small problem instances. Hence, we expect that solution quality will improve with QECCs.

Understanding the limitations of quantum algorithms for industry-relevant problems is still an open problem. It is important to not only hand-pick idealized problems, but to test the complexity of real data and real use cases. Moreover, industrial adoption of quantum technologies requires building a strong software infrastructure and (automated) toolchains. Although, specialized knowledge (*i.e.*, quantum physics) is still needed to program and make sense of results of quantum optimization, the entry barrier for practitioners is being continuously lowered by software development, which is reflected in the relative ease of use of tools such as those from vendors like D-Wave or IBM.

**Acknowledgments** This project was carried out within the TAQO-PAM consortium from the German Federal Ministry of Education and Research (BMBF) funding program “Quantum Technologies— from Basic Research to Market”, grant #13N15647, with funding codes #13N16092 (LS, WM), #13N16095 (VJ, US, TH) and

#13N16268 (CAR, FH). WM acknowledges support by the High-Tech Agenda Bavaria. Furthermore, we thank Martin Zehetmaier, Sebastian Ortlepp, and Carsten Tham, from BMW, and Georg Fraunhofer from OptWare GmbH for valuable discussions and proofreading the manuscript.

## REFERENCES

- [1] E. Farhi, J. Goldstone, and S. Gutmann, “A quantum approximate optimization algorithm,” *arXiv preprint arXiv:1411.4028*, 2014.
- [2] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang, “Obstacles to variational quantum optimization from symmetry protection,” *Phys. Rev. Lett.*, vol. 125, p. 260505, Dec 2020. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevLett.125.260505>
- [3] —, “Hybrid quantum-classical algorithms for approximate graph coloring,” *Quantum*, vol. 6, p. 678, Mar. 2022. [Online]. Available: <http://dx.doi.org/10.22331/q-2022-03-30-678>
- [4] J. A. Montanez-Barrera and K. Michielsen, “Towards a universal qaoa protocol: Evidence of a scaling advantage in solving some combinatorial optimization problems,” 2024. [Online]. Available: <https://arxiv.org/abs/2405.09169>
- [5] 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>
- [6] J. Wurtz and P. J. Love, “Counterdiabaticity and the quantum approximate optimization algorithm,” *Quantum*, vol. 6, p. 635, Jan. 2022. [Online]. Available: <https://doi.org/10.22331/q-2022-01-27-635>
- [7] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li *et al.*, “A review on quantum approximate optimization algorithm and its variants,” *Physics Reports*, vol. 1068, pp. 1–66, 2024, a review on Quantum Approximate Optimization Algorithm and its variants. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0370157324001078>
- [8] A. Lucas, “Ising formulations of many np problems,” *Frontiers in Physics*, vol. 2, 2014. [Online]. Available: <https://www.frontiersin.org/journals/physics/articles/10.3389/fphy.2014.00005>
- [9] A. Montanaro and L. Zhou, “Quantum speedups in solving near-symmetric optimization problems by low-depth qaoa,” 2024. [Online]. Available: <https://arxiv.org/abs/2411.04979>
- [10] R. Shaydulin, C. Li, S. Chakrabarti, M. DeCross, D. Herman *et al.*, “Evidence of scaling advantage for the quantum approximate optimization algorithm on a classically intractable problem,” *Science Advances*, vol. 10, no. 22, p. eadm6761, 2024. [Online]. Available: <https://www.science.org/doi/abs/10.1126/sciadv.adm6761>
- [11] N. Sachdeva, G. S. Hartnett, S. Maity, S. Marsh, Y. Wang *et al.*, “Quantum optimization using a 127-qubit gate-model ibm quantum computer can outperform quantum annealers for nontrivial binary optimization problems,” 2024. [Online]. Available: <https://arxiv.org/abs/2406.01743>
- [12] J. Preskill, “Quantum computing in the nisq era and beyond,” *Quantum*, vol. 2, p. 79, 2018. [Online]. Available: <https://doi.org/10.22331/q-2018-08-06-79>
- [13] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea *et al.*, “Noisy intermediate-scale quantum algorithms,” *Rev. Mod. Phys.*, vol. 94, p. 015004, Feb 2022. [Online]. Available: <https://link.aps.org/doi/10.1103/RevModPhys.94.015004>
- [14] M. X. Goemans and D. P. Williamson, “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming,” *J. ACM*, vol. 42, no. 6, p. 1115–1145, Nov. 1995. [Online]. Available: <https://doi.org/10.1145/227683.227684>
- [15] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006. [Online]. Available: <http://www.jstor.org/stable/j.ctt7s8xg>
- [16] D. M. Nenko and A. Caspari, “Dynamic optimization on quantum hardware: Feasibility for a process industry use case,” *Computers & Chemical Engineering*, vol. 186, p. 108704, 2024.
- [17] Z. Deng, X. Wang, and B. Dong, “Quantum computing for future real-time building hvac controls,” *Applied Energy*, vol. 334, p. 120621, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0306261922018785>
- [18] J. Fernández-Villaverde and I. Hull, “Dynamic programming on a quantum annealer: Solving the rbc model,” 2023. [Online]. Available: <https://arxiv.org/abs/2306.04285>

- [19] T. Häner, M. Roetteler, and K. M. Svore, "Optimizing quantum circuits for arithmetic," 2018. [Online]. Available: <https://arxiv.org/abs/1805.12445>
- [20] M. Schönberger, I. Trummer, and W. Maurer, "Quantum-inspired digital annealing for join ordering," in *Proceedings of the VLDB Endowment*, vol. 17, no. 3, 11 2023. [Online]. Available: <https://doi.org/10.14778/3632093.3632112>
- [21] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using qubo models," 2019. [Online]. Available: <https://arxiv.org/abs/1811.11538>
- [22] M. J. Schuetz, J. K. Brubaker, H. Montagu, Y. van Dijk, J. Klepsch *et al.*, "Optimization of robot-trajectory planning with nature-inspired and hybrid quantum algorithms," *Phys. Rev. Appl.*, vol. 18, p. 054045, Nov 2022. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevApplied.18.054045>
- [23] L. Schmidbauer, K. Wintersperger, E. Lobe, and W. Maurer, "Polynomial reduction methods and their impact on qaoa circuits," in *2024 IEEE International Conference on Quantum Software (QSW)*, vol. 986. IEEE, Jul. 2024, p. 35–45. [Online]. Available: <https://dx.doi.org/10.1109/QSW62656.2024.00018>
- [24] 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: <https://dx.doi.org/10.1088/1361-6633/ab85b8>
- [25] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, "Quantum annealing for industry applications: introduction and review," *Reports on Progress in Physics*, vol. 85, no. 10, p. 104001, sep 2022. [Online]. Available: <https://dx.doi.org/10.1088/1361-6633/ac8c54>
- [26] M. Vandelli, A. Lignarolo, C. Cavazzoni, and D. Dragoni, "Evaluating the practicality of quantum optimization algorithms for prototypical industrial applications," *Quantum Information Processing*, vol. 23, no. 10, p. 344, Oct 2024. [Online]. Available: <https://doi.org/10.1007/s11128-024-04560-1>
- [27] A. M. Krol, M. Erdmann, R. Mishra, P. Singkanipa, E. Munro *et al.*, "Qiss: Quantum industrial shift scheduling algorithm," 2024. [Online]. Available: <https://arxiv.org/abs/2401.07763>
- [28] M. E. S. Morales, J. D. Biamonte, and Z. Zimborás, "On the universality of the quantum approximate optimization algorithm," *Quantum Information Processing*, vol. 19, no. 9, Aug. 2020. [Online]. Available: <http://dx.doi.org/10.1007/s11128-020-02748-9>
- [29] J. A. Montanez-Barrera and K. Michielsen, "Towards a universal qaoa protocol: Evidence of a scaling advantage in solving some combinatorial optimization problems," 2024. [Online]. Available: <https://arxiv.org/abs/2405.09169>
- [30] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd *et al.*, "Adiabatic quantum computation is equivalent to standard quantum computation," *SIAM Rev. Soc. Ind. Appl. Math.*, vol. 50, no. 4, pp. 755–787, Jan. 2008.
- [31] C. C. McGeoch, *Adiabatic quantum computation and quantum annealing*, ser. Synthesis lectures on quantum computing. Cham: Springer International Publishing, 2014.
- [32] V. Choi, "The effects of the problem hamiltonian parameters on the minimum spectral gap in adiabatic quantum optimization," *Quantum Inf. Process.*, vol. 19, no. 3, Mar. 2020.
- [33] A. Braida and S. Martiel, "Anti-crossings and spectral gap during quantum adiabatic evolution," *Quantum Inf. Process.*, vol. 20, no. 8, Aug. 2021.
- [34] T. Fujii, K. Komuro, Y. Okudaira, and M. Sawada, "Eigenvalue-invariant transformation of ising problem for anti-crossing mitigation in quantum annealing," *J. Phys. Soc. Jpn.*, vol. 92, no. 4, Apr. 2023.
- [35] T. Zaborniak and R. de Sousa, "Benchmarking hamiltonian noise in the d-wave quantum annealer," *IEEE Transactions on Quantum Engineering*, vol. 2, p. 1–6, 2021. [Online]. Available: <http://dx.doi.org/10.1109/TQE.2021.3050449>
- [36] T. Imoto, Y. Susa, R. Miyazaki, T. Kadowaki, and Y. Matsuzaki, "Universal quantum computation using quantum annealing with the transverse-field ising hamiltonian," 2024. [Online]. Available: <https://arxiv.org/abs/2402.19114>
- [37] A. G. Nikolaev and S. H. Jacobson, "Simulated annealing," in *International Series in Operations Research & Management Science*, ser. International series in operations research & management science. Boston, MA: Springer US, 2010, pp. 1–39.
- [38] K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz *et al.*, "Enabling research through the scip optimization suite 8.0," *ACM Trans. Math. Softw.*, vol. 49, no. 2, jun 2023. [Online]. Available: <https://doi.org/10.1145/3585516>
- [39] L. Perron and V. Furnon, "Or-tools," Google. [Online]. Available: <https://developers.google.com/optimization/>
- [40] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman *et al.*, "Quantum computing with Qiskit," 2024. [Online]. Available: <https://arxiv.org/abs/2405.08810>
- [41] A. Broadbent and E. Kashefi, "Parallelizing quantum circuits," *Theoretical Computer Science*, vol. 410, no. 26, p. 2489–2510, Jun. 2009. [Online]. Available: <http://dx.doi.org/10.1016/j.tcs.2008.12.046>
- [42] R. Diestel, *Graph Theory*, 3rd ed. Springer-Verlag Heidelberg, New York, 2005. [Online]. Available: <http://www.math.ubc.ca/~solymosi/2007/443/GraphTheoryIII.pdf>
- [43] J. Kattemölle, "Edge coloring lattice graphs," 2024. [Online]. Available: <https://arxiv.org/abs/2402.08752>
- [44] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg *et al.*, "Evidence for the utility of quantum computing before fault tolerance," *Nature*, vol. 618, no. 7965, pp. 500–505, Jun. 2023.
- [45] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman *et al.*, "Quantum computing with Qiskit," 2024.
- [46] N. Dattani, S. Szalay, and N. Chancellor, "Pegasus: The second connectivity graph for large-scale quantum annealing hardware," 2019. [Online]. Available: <https://arxiv.org/abs/1901.07636>
- [47] J. Cai, W. G. Macready, and A. Roy, "A practical heuristic for finding graph minors," 2014. [Online]. Available: <https://arxiv.org/abs/1406.2741>
- [48] Z. Dvořák, *Graph minors*. Cham, Switzerland: Springer International Publishing, May 2025.
- [49] S. Ball, A. Centelles, and F. Huber, "Quantum error-correcting codes and their geometries," *Annales de l'Institut Henri Poincaré D, Combinatorics, Physics and their Interactions*, vol. 10, no. 2, p. 337–405, Feb. 2023. [Online]. Available: <http://dx.doi.org/10.4171/aihpd/160>
- [50] E. B. Niven and C. V. Deutsch, "Calculating a robust correlation coefficient and quantifying its uncertainty," *Computers and Geosciences*, vol. 40, p. 1–9, Mar. 2012. [Online]. Available: <http://dx.doi.org/10.1016/j.cageo.2011.06.021>
- [51] V. Shende, S. Bullock, and I. Markov, "Synthesis of quantum-logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 1000–1010, 2006.
- [52] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 6 2012.
- [53] S. Zbinden, A. Bäertschi, H. Djidjev, and S. Eidenbenz, *Embedding Algorithms for Quantum Annealers with Chimera and Pegasus Connection Topologies*. Springer International Publishing, 2020, p. 187–206. [Online]. Available: [http://dx.doi.org/10.1007/978-3-030-50743-5\\_10](http://dx.doi.org/10.1007/978-3-030-50743-5_10)