Ready to Leap (by Co-Design)? Join Order Optimisation on Quantum Hardware

Manuel Schönberger
Technical University of Applied Sciences Regensburg
Regensburg, Germany
manuel.schoenberger@othr.de

Stefanie Scherzinger
University of Passau
Passau, Germany
stefanie.scherzinger@uni-passau.de

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

ABSTRACT

The prospect of achieving computational speedups by exploiting quantum phenomena makes the use of quantum processing units (QPUs) attractive for many algorithmic database problems. Query optimisation, which concerns problems that typically need to explore large search spaces, seems like an ideal match for the known quantum algorithms. We present the first quantum implementation of join ordering, which is one of the most investigated and fundamental query optimisation problems, based on a reformulation to quadratic binary unconstrained optimisation problems.

Current QPUs are classified as noisy, intermediate scale quantum computers (NISQ), and are restricted by a variety of limitations that reduce their capabilities as compared to ideal future quantum computers, which prevents us from scaling up problem dimensions and reaching practical utility. To overcome these challenges, our formulation accounts for specific QPU properties and limitations, and allows us to trade between solution quality and problem size.

We empirically characterise our method on two state-of-the-art NISQ approaches (gate-based quantum computing and quantum annealing), and confirm that technological limits are quickly reached. In contrast to prior work on quantum computing for query optimisation, we go beyond currently available QPUs, and explicitly target the scalability limitations: Using insights gained from numerical simulations and our experimental analysis, we identify key criteria for co-designing QPUs to improve their usefulness for join ordering, and show how even relatively minor physical architectural improvements can result in substantial enhancements. Finally, we outline a path towards practical utility of custom-designed QPUs, which we envision as local query optimisation accelerators.

PVLDB Reference Format:
Manuel Schönberger, Stefanie Scherzinger, and Wolfgang Mauerer. Ready to Leap (by Co-Design)? Join Order Optimisation on Quantum Hardware. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/lfd/VLDB23/.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

1 INTRODUCTION

In recent years, quantum computing has attracted substantial attention in many fields of research, driven by the desire to benefit from quantum advantage in complex computations. While quantum computing has been studied for decades, the increase in interest aligns with the accelerating development of quantum computing hardware over the recent years, provided by vendors like as IBM [39], Rigetti [69], or D-Wave [53], among many others. Moreover, cloud access to quantum systems has made quantum computing more accessible to researchers, enabling first experiments on real *quantum processing units* (QPU).

In contrast to CPUs, QPUs work with quantum bits, or *qubits*. Their mathematical state is exponentially larger than for classical bits, and they can realise phenomena like *quantum superposition*, *quantum entanglement* or *quantum interference* [60]. It is widely believed, given accepted complexity theoretic assumptions, that quantum systems offer increased computational power over classical systems [5, 8]. Speedups have been proven for multiple quantum algorithms [31, 74], and a seminal experiment [6] has demonstrated quantum advantage on real hardware, even if on an artificially constructed problem.

Quantum computers (QCs) are also believed to excel at optimisation problems that need to determine elements with specific properties in (exponentially large) search spaces, which is a commonly occurring problem in database systems, and particularly relevant for database query optimisation. However, QCs have so far seen only meagre adoption in DB research, and multi query optimisation (MQO) [23, 73, 79] is the only application in query optimisation that we are aware of, to the best of our knowledge. In this paper, we investigate the aptitude of quantum computing for the classic join ordering (JO) problem, which is one of the most extensively researched and fundamental problems in the field of query optimisation [47, 55, 58, 59, 76, 80, 86].

However, using state-of-the-art quantum systems comes with several challenges. It is not possible to simply deploy existing classical algorithms on QPUs, and problems must instead be addressed using custom quantum algorithms, or by reformulating them into certain mathematical descriptions that are unaccustomed in traditional programming. Moreover, the reformulation needs to consider the (many) limitations of current QPUs, such as limited quantum coherence time [68] that restricts the time interval during which a QPU functions properly for a computation, or the limited amount of qubits offered by current QPUs that necessitates efficient problem formulations in not just the overall resource scaling behaviour, but also in exact details that are usually ignored in complexity analysis.

These limitations prevent current QPUs, which are classified as so-called *noisy intermediate scale quantum* (NISQ) devices [65], from solving problems of practically relevant dimensions. At the given time, we do not expect meaningful results for practical JO problems. Instead, we propose a reformulation of the problem that accounts for such limitations, provide a formal analysis of bounds, experimentally analyse the achievable performance on multiple NISQ machines, and suggest physical improvements for future QPUs that benefit query optimisation workloads.

Contributions. In detail, our contributions are as follows:

(1) We show how join ordering problems can be solved on QPUs, providing novel mathematical reformulations that account for their specific properties and restrictions.
(2) We conduct an extensive experimental analysis for join ordering on real QPUs, where we comprehensively consider two state-of-the-art architectures, and show that fundamental limits are quickly reached when we scale up problem dimensions. We moreover show how specific parameter configurations, that provide no issues in a classical context, have a major impact on the feasibility of using QPUs for join ordering.
(3) To address problem scalability, we formally derive an upper bound for qubit resource scaling that quantifies all influences.
(4) We specify recommendations for DB-QPU co-design, which has so far not been addressed by prior research. We combine insights from numerical simulations and our experimental analysis, and show how even small architectural improvements can substantially enhance the feasibility of using QPUs for join ordering. This paves a way towards practical utility of QPUs in DB applications, which we envision as local co-processors for accelerating query processing as illustrated in Fig. 1.
(5) We provide a fully reproducible implementation of our analysis and problem reformulation. The latter may serve as a basis for future extensions.

The rest of the paper is structured as follows: We give a very brief overview on quantum computing foundations in Sec. 2. Considering architectural limitations of QPUs, we present our approach for implementing join ordering on QPUs in Sec. 3. We then experimentally evaluate our approach on early-stage QPUs in Sec. 4. In Sec. 5, we formally derive an upper bound for the number of qubits required to encode JO problem. In Sec. 6, we analyse extensions required for future QPUs to solve practical join ordering problems. After presenting related work in Sec. 7, we conclude in Sec. 8.

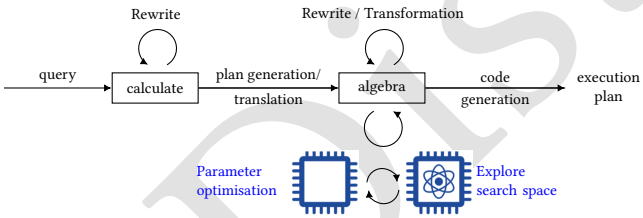

Figure 1: Envisioning a QPU as a co-processor in database query optimization (adapting from [55]).

2 QUANTUM FOUNDATIONS

Quantum computing is a relatively new computational paradigm, and prototypical hardware has only recently become available. It is impossible to provide a complete introduction to the field here, and we refer to Nielsen and Chuang [60] as seminal textbook, and Bharti et al. [9] as algorithmic review instead. Nonetheless, since the approach differs greatly from the customary hardware foundations, we need to lay the ground for our considerations.

2.1 Quantum Algorithms

Existing classical algorithms cannot be trivially converted to quantum systems. Instead, a quantum algorithm that computes a quantum state representing an optimal or near-optimal solution to a problem with high probability is required. These algorithms fall, broadly speaking, into two categories: Algorithms with provable speedups over their classical counterparts that require a fully functioning, error corrected quantum computer to work, and heuristic approaches working on hardware that is physically realisable now and in the medium term future.

The first category includes, for instance, Shor's seminal factoring algorithm, or Grover search in large, unstructured spaces.[1] The second class contains heuristic approaches that are believed to optimally use quantum phenomena in non-error corrected, present-day quantum computers, so-called noisy intermediate scale quantum (NISQ) machines. Except for notable efforts like supremacy experiments on an artificially constructed problem [6] or a kernel-based supervised machine learning approach [49], algorithms in this class do not enjoy a strict proof of computational advantage, or even currently experimentally verified advantages on available hardware.

The class mostly comprises *hybrid variational quantum algorithms*: parameterised quantum gates produce quantum states (using superposition and entanglement) that are supposed to, roughly speaking, explore scenarios faster than classical counterparts. Their believed speedups rest on quantum interference phenomena for gate-based approaches, and on quantum fluctuations for quantum annealing. Detailed rationales on why speedups are expected, and the physical intuition behind, cannot be discussed here for the lack of space. Instead, we refer to Bharti *et al.* [9] for gate-based systems, and Albash and Lidar [3] for quantum annealers.

2.2 QPU Architectures

While physical implementation techniques for quantum computers vary widely, two conceptual paradigms have emerged that we consider in this work: Gate-based QPUs and quantum annealers. We outline their essential characteristics, and show the steps required to solve the JO problem.

2.2.1 Gate-based QPUs. Multiple vendors (*e.g.*, IBM, AQT, IQM, Rigetti) offer access to early commercial hardware.

Computation Model. Quantum gates operate (conceptually similar to electronic gates) on the quantum state represented by a

[1] Grover's algorithm was initially misnamed as *database search*, albeit the meaning of database does not match what is considered a typical database: The algorithm works on *conceptual* search spaces, not physical data, unless these can be stored in quantum RAM, whose implementation in quantities even beyond a few quantum bits only is still a major physical challenge [9],

collection of qubits. Gates form a *quantum circuit* that implements a computation, and forms part of an algorithm.

A single quantum state can be read by a measurement, which is typically performed at the end of a circuit. Measurements (very roughly speaking [64]) collapse a quantum state into *one of multiple possible* classical states. The outcome follows a probability distribution that depends on quantum state produced by the circuit. A common algorithmic pattern is to perform operations on an input state such that a favourable state, representing a solution to a problem, is obtained with high probability. Typically, circuits are not just run once, but instead iterate over tens to thousands of *shots* to obtain a statistical distribution.

Gate-Based Quantum Optimisation. Our considerations for gate-based QPUs rest on the *quantum approximate optimisation algorithm* (QAOA) [24] method. It is an iteration-based, hybrid quantum-classical algorithm. Each QAOA iteration has (1) a quantum step, in which a parameterised quantum circuit explores the search space using quantum states, followed by a measurement, and (2) a classical step to tune the circuit parameters with a classical numerical optimiser, based on the measurement results. The optimised parameters are used for the next iteration. The quantum circuit consists of alternating a *cost* and a *mixing* operator. The cost operator encodes the optimisation problem to be solved, and the mixing operator ensures that the quantum state appropriately explores the search space. The quantum sequence is of length $2p$, where p denotes the number repetitions of cost and mixing operator. As shown by Farhi *et al.* [24], the approximation quality improves as p increases. However, even for $p = 1$, QAOA shows promising experimental results for some problems [24, 25], and it is know that an efficient classical algorithm for sampling the output distribution of QAOA with $p = 1$ would imply a collapse of the polynomial hierarchy [27], which is seen as clear indicator of possible quantum advantages. It is still an open research question if and when *practical* benefits can be obtained on current hardware. Optimisation (sub-)problems that can be mapped to QAOA appear frequently in databases, and often do not concern the payload itself[2]—instead, they address *conceptual* search spaces, for instance for join order optimisation, and are therefore promising candidates for quantum speedups.

Problem Encoding. QAOA requires problems to be encoded as an *Ising Hamiltonian* [24], which is (for our purposes) equivalent to a *quadratic unconstrained binary optimization* (QUBO) problem formulation [10, 48] based on the multivariate polynomial

$$f(\vec{x}) = \sum_i c_{ii} x_i + \sum_{i \neq j} c_{ij} x_i x_j, \tag{1}$$

where $x_i \in \{0, 1\}$ are variables, and $c_{i,j} \in \mathbb{R}$ coefficients (it holds that $c_{ij} = c_{ji}$). Note that Eq. (1) can be interpreted as weighted, undirected graph with adjacency matrix given by the coefficients $c_{i,j}$. The QAOA algorithm seeks to determine $\arg\min_{\vec{x}} f(\vec{x})$. Note that QUBOs do not allow for specifying explicit constraints (for instance, like in linear programs), and can only express pairwise interactions between variables.

[2]Quantum RAM is extremely hard to manufacture, and current technology allows for producing a few qubits at most [9], which will likely remain a challenge in the near- and mid-term future. Loading even parts of the payload into quantum resources to work quantum algorithms on the actual data is therefore not a viable path.

Problems must be cast into QUBO form such that the minimum value corresponds to a valid and optimal solution (this is possible for all NP-complete problems [50]). To the best of our knowledge, no QUBO formulations have yet been proposed for JO. One particular challenge is to find a QUBO reformulation of problems that aligns well with the QPU properties. For instance, more quadratic QUBO contributions result in deeper QAOA circuits [24], which are therefore subject to larger degradation by current-day technical imperfections. In Sec. 3, we propose a suitable encoding considering such constraints.

QPU Embedding/Transpilation. After building the QAOA circuit based on the problem formulation, it must be *embedded* onto the QPU to match the hardware constraints regarding qubit topology and available gates [14]. The logical circuit is transpiled into a functionally equivalent circuit that only uses gate operations included in the *native QPU gateset*. Also limited interaction possibilities between qubits must be accounted for by inserting appropriate *swap* gates [14] when non-adjacent qubits are supposed to interact with each other, which increases circuit depth.

QPU Limitations. The execution of a physical circuit on NISQ hardware is subject to various perturbations that decrease result quality. Most importantly, current QPUs feature a limited coherence time [60]: Increasing quantum circuit depth (*i.e.*, the longest sequence of operators in a circuit) increases the occurrence probability of errors because of, for instance, a loss of quantum information due to interactions with the environment [68]. The coherence times $T1$ and $T2$ quantify the speed of decoherence [57, 77]. The probability of *gate errors* likewise increases with increasing gate count. Consequently, it is important to find formulations of limited size.

2.2.2 Quantum Annealers. Quantum annealers (as, for instance, offered by D-Wave [53]) implement a restricted variant of adiabatic quantum computing (the general form is known to be equivalent to gate-based quantum computing [2]) on NISQ hardware [26, 65]. It is subject to debate whether or not quantum annealing (QA) can achieve speedups over classical systems [65].

Currently available quantum annealers offer thousands of qubits, compared to hundreds of qubits in gate-based systems. They only allow for interactions between a restricted set of qubit pairs, as given by their connectivity graph.

Computation Model. QA also seeks to determine the minimum energy, or ground state, of a problem Hamiltonian [26]. The same QUBO encoding as for gate-based QPUs is applicable to quantum annealers. QA can find minimum energy solutions to QUBOs, but is incapable of running other quantum algorithms.

Contrary to gate-based systems, where computation time is determined by the depth of a circuit and the types of gates, runtime is a parameter for annealers: If it is too small, a non-optimal solution will result; if it is too large, this is obviously counter to speedups. The minimally required annealing time is (roughly, and ignoring many details) inversely proportional to the minimum energy gap between the ground and first excited state of the Hamiltonian [26]. This eigenvalue gap depends on the chosen encoding, and is, as a quantum many-body problem, difficult to compute [3]. Additionally, pairwise interactions between variables must be mapped to the possibilities of physical hardware. Consequently, the concrete

encoding of problems into QUBO form is also relevant for QA, and determining the possible instance size of JO problems on quantum annealers is a non-trivial question that we address.

QPU Embedding. To minimise a logical QUBO on physical hardware, we need to map the QUBO graph onto the connectivity graph of the QA. Variables are mapped to qubits, and coefficients $c_{i,j}$ in Eq. (1) determine the physical coupling strengths between qubits. In the D-Wave Advantage system, every physical qubit is connected to a limited set of 15 other qubits following a certain topological pattern (Pegasus graph [53]). Consequently, not every possible interaction $c_{i,j}$ can be directly realised on hardware.

To represent such interactions, sets of connected physical qubits are combined into qubit chains [13] so that not directly connected qubits can be made to interact. A larger mismatch between QUBO and QA hardware graph leads to more and longer such chains. This substantially reduces the usable amount of qubits, and increases the error probability during the annealing process [53].

Embedding QUBO onto a given QPU topology had to be performed by hand in first applications of QA on DB problems [79], which is possible for specific problems with predictable connectivity requirements following fixed patterns (like MQO), but becomes infeasible for more dynamic problems like JO. Additionally, the problem is NP-complete [51, 87], and any manual approach quickly becomes infeasible. Approximative or heuristic techniques (see, *e.g.*, Refs. [22, 87]) are available; yet, we find that reducing the complexity of the QUBO formulation remains a crucial requirement.

3 JOIN ORDERING ON QPUS

While casting JO as QUBO is obviously possible, it is crucial to find a formulation that is well adapted to the constraints of QPUs. This is a common hurdle in quantum computing, as properties of mathematically equivalent formulations substantially impact both, feasibility (can the problem be mapped to a QPU of given resources?) and quality (how susceptible is the formulation to imperfections?). For JO, we are unaware of any previous QUBO formulation. Our approach builds on the following steps:

(1) Express JO as *mixed integer linear programming (MILP)* problem,
(2) Carefully adjust the MILP formulation to form a *binary integer linear programming (BILP)* model,
(3) Transform BILP into QUBO, suitable for QPU processing.

3.1 Join Ordering with MILP

We first reformulate JO as a MILP problem, inspired by a contribution by Trummer and Koch [80] for classical solvers (and unrelated to their work on multi-query optimisation on QPUs). The approach seeks optimal left-deep join trees that allow for cross products, without restrictions on the query graph (such JO problems are known to be NP-complete [19]).

Solving a MILP problem entails determining a value assignment for variables of integer or continuous domains, such that a given linear objective function is optimised [20]. For valid solutions, the variables need to satisfy a given set of linear constraints.

For execution on QPUs, we need to nontrivially adapt the MILP encoding of Ref. [79], such that it satisfies constraints not present in the original setting: The original model introduces additional variables to improve comprehensibility, relying on redundant variable elimination by the solver. Since additional variables non-linearly translate to additional qubits, we need to reduce the model up-front to retain feasibility on restricted hardware. Nevertheless, the class of JO problems considered remains identical to the original MILP reformulation.

In [80], it is shown how to model cost functions for a variety of operators, such as hash join and sort-merge join. However, these require additional variables. Since we require one qubit for each variable, as we will show in Section 3.4, and since we seek to keep the number of required qubits at a minimum, we only consider the more classic function C_{out} [19], which is given by $C_{out}(n_i, n_j) := n_i n_j f_{i,j}$, where n_i and n_j are the cardinalities of relations R_i and R_j to be joined and $f_{i,j}$ is the join selectivity. Following Cluet and Moerkotte [19] to find the optimal join order for a sequence s of n relations $s_1, ..., s_n$, the cost function becomes

$$C(s) := \sum_{i=2}^{n} C_{out}(|s_1 ... s_{i-1}|, |s_i|), \tag{2}$$

where $|s_1 ... s_{i-1}|$ denotes the cardinality of the intermediate join result after joining $s_1, ..., s_{i-1}$. Directly encoding this cost function into the MILP objective function is not possible because the required product operations in C_{out} cannot be represented in the linear objective function with linear constraints.

To circumvent this issue, Trummer and Koch [80] propose to use logarithmic cardinalities, as $\log(\prod_i a_i) = \sum_i \log a_i$. Cardinalities are approximated via an arbitrary number of threshold variables. For quantum formulations, this results in a trade-off between better approximation and more qubits that requires great care.

3.2 Pruning the MILP Model

As each binary variable in our model requires one qubit, we need to reduce the number of variables. Modern MILP solvers can, to some extent, detect and prune redundancies [1]. However, as model size is crucial on QPUs, we manually ensure their removal. Additionally, precise knowledge of the influence of variables and constraints is important for the formal analysis in Sec. 5

Modelling Relations. We follow approach and naming conventions of Ref. [80]. For each of the T relations and J joins, we distinguish between inner and outer operand in the left-deep join tree (outer operands are the result of preceding joins). The binary variables tii_{tj} (*Table In Inner join operand*) and tio_{tj} (*Table In Outer join operand*) indicate whether relation t with $0 \leq t \leq T, t \in \mathbb{N}_0$ is part of the inner or outer operand of join j with $0 \leq j \leq J, j \in \mathbb{N}_0$. We add $2TJ$ such variables to the model. To enforce solution validity, constraints $\sum_t tii_{tj} = 1$, added for each join j, and $\sum_t tio_{t0} = 1$ together ensure that each leaf of the join tree corresponds to exactly one relation. For each join $j > 0$ and relation t, the constraint

$$tio_{tj} = tii_{t,j-1} + tio_{t,j-1} \tag{3}$$

enforces that a relation will be part of the outer operands of all subsequent joins once it is initially included in a join. Additional constraints ensure that the same relation cannot be part of both the inner and outer operand of the join. For all except the final join, these constraints are accounted for by the constraints in Eq. (3), and therefore redundant. It suffices to include constraints for the

final join and for each relation t:

$$tio_{t,J-1} + tii_{t,J-1} \leq 1. \tag{4}$$

The join order representing a solution is given by all variables tio_{t0} and tii_{tj} that are set to 1, that is, the leaf nodes of the join tree.

Example 3.1. *For each $0 \leq t \leq 2$ representing relations R, S and T, and for each join $0 \leq j \leq 1$, we introduce tio_{tj} and tii_{tj}. The constraints enforce valid assignments, so either tio_{00}, tio_{10} or tio_{20} must be set to 1, whereas the two remaining variables must equal 0, since exactly one relation must represent a join tree leaf. The same holds for the remaining leaf variables tii_{t0} and tii_{t1} for each relation t.*

Assume that $tio_{00} = 1$ and $tii_{10} = 1$, encoding join 0 as $R \bowtie S$. Then, $tio_{01} = 1$ and $tio_{11} = 1$, as both R and S are constrained to be included in the outer operand of join 1 as a result of join 0. This implies $tii_{21} = 1$ and therefore relation T as the inner operand for join 1, since each leaf must be represented by one relation and no relation can be part of both the inner and outer operand of the same join. Remaining variables are constrained to 0, yielding join order $(R \bowtie S) \bowtie T$.

Modelling Predicates. Consider binary predicates for joining two relations.[3] For each predicate p with $0 \leq p \leq P, p \in \mathbb{N}_0$, where P denotes the overall number of predicates, and for each join $j > 0$, we introduce a variable pao_{pj} (*Predicate Applicable in Outer join operand*). It indicates whether predicate p can be evaluated for the outer operand of join j, so *both* associated relations are part of the outer operand of join j. For predicate p and join $j > 0$, this is enforced by the constraints

$$pao_{pj} \leq tio_{T_1(p)j}, \qquad pao_{pj} \leq tio_{T_2(p)j}, \tag{5}$$

where $T_1(p)$ and $T_2(p)$ denote the first and second relation for predicate p. The original model includes variables pao_{p0}. However, we prune these, since the outer operand of the very first join contains only a single relation. In total, we add $P(J-1)$ many pao_{pj} variables.

Example 3.2. *(cont'd) Consider the inclusion of join predicate p_{RS} for relations R and S. One additional variable pao_{01} is required to denoting whether p_{RS} (indexed by 0) can be applied for join 1. The pruned model omits the superfluous variable pao_{00} for join 0.*

Two constraints $pao_{01} \leq tio_{01}$ and $pao_{01} \leq tio_{11}$ enforce that pao_{01} may only be set to one if both R and S are included in the outer operand of join 1, which is the case for $(R \bowtie S) \bowtie T$. The predicate may therefore be applied, which impacts the cardinality calculation. Similarly, we may add a predicate p_{RT} or p_{ST} for the relation T. If no predicate is provided, this necessitates a cross product for T.

Cost Function and Cardinality Approximation. Estimating intermediate cardinalities can be encoded as a MILP problem, based on a logarithmic representaton [80]. c_j denotes the logarithmic cardinality for the outer operand of join j by

$$c_j = \sum_t \log(Card(t)) tio_{tj} + \sum_p \log(Sel(p)) pao_{pj}, \tag{6}$$

where $Card(t) \geq 1$ is the cardinality of relation t, and $0 < Sel(p) \leq 1$ is the selectivity of predicate p. The cardinalities for each outer join operand are approximated using R threshold values. Following Ref. [80], for each threshold value r with $0 \leq r \leq R, r \in \mathbb{N}_0$ and each join j, a variable cto_{rj} (*Cardinality Threshold reached by Outer operand*) is added to indicates if the intermediate logarithmic cardinality for the outer operand of join j exceeds the threshold value r. If $cto_{rj} = 1$, the threshold value is added to the objective function $\min \sum_{r=0}^{R-1} \sum_{j=1}^{J-1} cto_{rj}\theta_r$, where θ_r denotes the r-th threshold value. Since we use the cost function outlined in Equation 2 and only consider intermediate cardinalities, we prune the variables cto_{r0}. Therefore, $R(J-1)$ variables of type cto_{rj} are required. This is an upper bound, since some cases allow us to prune further variables.

To ensure that variables are assigned correct values,

$$c_j - cto_{rj} \cdot \infty_{rj} \leq \log(\theta_r). \tag{7}$$

enforces that cto_{rj} is activated if the logarithmic cardinality c_j exceeds the threshold value, since the inequality can then only be satisfied by setting $cto_{rj} = 1$, thereby subtracting the (sufficiently large) constant ∞_{rj} from the left-hand side of the inequality. Contrary to the original model, c_j is not included, but is merely used for convenience, abbreviating the calculation shown in Eq. (6).

We observe that in Eq. (7), variable cto_{rj} can be pruned if the maximum value of the logarithmic intermediate cardinality for outer operand of join j (which we specify in Lemma 5.2) does not exceed $\log(\theta_r)$. This may occur for large θ_r and early joins. In these cases, subtracting $cto_{rj} \cdot \infty_{rj}$ is never required to satisfy the constraint, rendering both variable and constraint obsolete.

Example 3.3. *(cont'd) Consider the simple scenario of $Card(R) = Card(S) = Card(T) = 100$ to specify input cardinalities. Let $Sel(p_{RS}) = 0.1$. Clearly, optimal join orders are $(R \bowtie S) \bowtie T$ and $(S \bowtie R) \bowtie T$, where the actual costs for the intermediate join are given by $Card(R) \cdot Card(S) \cdot Sel(p_{RS}) = 1{,}000$. However, for the MILP approach, we must approximate this intermediate cardinality using threshold values, which we assume to be $\theta_0 = 100$ and $\theta_1 = 1{,}000$. We add variables cto_{01} and cto_{11} for each of the thresholds and for join 1, which has the intermediate result $R \bowtie S$ as the outer operand. As we use the cost function outlined in Eq. (2) and therefore only consider intermediate results, we do not add variables for join 0, which has the input relation R as an outer operand. For both variables, we add a constraint as given in Eq. (7). For c_j, we obtain $c_j = \log(Card(R)) + \log(Card(S)) + \log(Sel(p_{RS})) = 3$. Since $3 > \log(\theta_0) = \log(100) = 2$, $cto_{01} = 1$ to satisfy the constraint. However, since $3 \leq \log(\theta_1) = \log(1{,}000) = 3$, $cto_{01} = 0$ satisfies the constraint. Therefore, only $\theta_0 = 100$ is added to the costs, which is far from the actual intermediate cardinality for $R \bowtie S$. Evidently, the choice of threshold values greatly impacts the accuracy. This requires careful consideration for QPUs, where finding a balance between sufficient accuracy and qubit count is crucial.*

Table 1 summarises savings in variables and constraints by pruning the MILP model. As we discuss below, this is crucial since both substantially impact the number of required qubits, and decisively influences the feasibility on current NISQ machines.

3.3 BILP Formulation

To transformation from MILP to QUBO, as required by QPUs, we build on an intermediate BILP [4] step, since efficient transformations

[3] We restrict our consideration to uncorrelated predicates for lower qubit requirements, but an extension of the model to correlated predicates is discussed in Ref. [80].

[4] Given a vector of n binary variables $x \in \{0,1\}^n$ and a cost vector $c \in \mathbb{R}^n$, an *optimal* solution assigns variables to minimise $c \cdot x$ and adheres to to m constraints

Table 1: Comparison of variables and constraints n in pruned and original [80] MILP model.

Constraint Type	Original Model	Pruned Model
$tio_{tj} + tii_{tj} \leq 1$	$n = TJ$	$n = T$
$pao_{pj} \leq tio_{T_1(p)j}$	$n = PJ$	$n = P(J-1)$
$pao_{pj} \leq tio_{T_2(p)j}$	$n = PJ$	$n = P(J-1)$
$c_j - cto_{rj} \cdot \infty_{rj} \leq \log(\theta_r)$	$n = RJ$	$n \leq R(J-1)$
Variable Type	**Original Model**	**Pruned Model**
pao_{pj}	$n = PJ$	$n = P(J-1)$
cto_{rj}	$n = RJ$	$n \leq R(J-1)$

from BILP to QUBO are known [50], at least for problems restricted to binary variables and equality constraints. The pruned MILP model includes inequality constraints that we turn into equality constraints by adding slack variables [20]. For instance, converting the constraint in Eq. (7) by adding a slack variable s_{rj} gives

$$c_j - cto_{rj} \cdot \infty_{rj} + s_{rj} = \log(\theta_r). \tag{8}$$

However, this violates the restriction to binary variables, since s_{rj} needs to be continuous. We therefore approximate s_{rj} by multiple *binary* slack variables, since an integer bounded by C can be expressed using $n = \lfloor \log_2(C) \rfloor + 1$ binary variables [16]. This gives $s_{rj} \approx \omega \sum_{i=1}^{n} 2^{i-1} b_i$, where ω denotes the approximation precision, and $b_i \in \{0, 1\}$. This results in

$$n = \lfloor \log_2 (C/\omega) \rfloor + 1 \tag{9}$$

binary variables, which, again, leads to a trade-off: More binary variables for the approximation (smaller ω) lead to higher precision, which is costly given the limited number of qubits. However, without any remaining inequality constraints, the BILP problem can now be trivially cast as QUBO suitable for QPUs.

3.4 QUBO Formulation

Encoding. Unlike with BILP, QUBO problems cannot include explicit constraints. Instead, we need to ensure that a solution with minimum value inherently corresponds to a valid solution. Lukas [50] provides a conversion that turns BILP problems into QUBOs of the form

$$H = H_A + H_B = \underbrace{A \sum_{j=1}^{m} \left(b_j - \sum_{i=1}^{N} S_{ji} x_i \right)^2}_{H_A} + \underbrace{B \sum_{i=1}^{N} c_i x_i}_{H_B}, \tag{10}$$

where H_A ensures valid, and H_B optimal solutions.

H_A encodes the BILP constraints $Sx = b$. The inner quadratic term evaluate to 0 iff no constraint is violated. Invalid solutions are penalised by A, and cannot correspond to the minimum value.

Through the discretisation of continuous variables, $b_j - \sum_{i=1}^{N} S_{ji} x_i$ may only be close, but not equal to 0 even for valid solutions. To circumvent this problem, we round the coefficients S_{ji} according to the discretisation precision ω.

Term H_B encodes the cost. Weights A and B in Eqn. (10) must be suitably assigned. Since we prioritise valid and non-optimal solutions over optimal but invalid ones, $A \gg B$ needs to hold. The weights cannot be set to arbitrarily large values, as quantum annealers have limited resolutions to tune couplings, and high penalty weights are also known to cause issues like slowdowns [61].

We choose the smallest possible weights such that violating a single constraint by the smallest possible amount already leads to a sufficiently large penalty such that the solution cannot correspond to the minimum energy.

To determine this smallest possible violation, first consider constraints with only binary variables. The minimum violation is 1 in this case—for instance, in constraint $1 - x_1 + x_2 = 0$, with binary variables, the configuration $x_1 = x_2 = 1$ leads to a violation by 1, and contributes penalty $A \cdot 1$. In contrast, constraint $c - 1.1 = 0$ with continuous variable c discretised at precision $\omega = 0.1$ delivers a minimal violation for $c = 1.2$ or $c = 1.0$, contributing a penalty of $A(0.1)^2 = A(\omega)^2$. Therefore, the smallest violation is given by ω, and hence, $A = C/\omega^2 + \epsilon$ for $B = 1$, where ϵ is some small value and $C = \sum_{i=1}^{N} c_i$. Violating a constraint to save costs that would otherwise be added in H_B is discouraged, as it will always lead to the same or even larger costs.

Quadratic Contributions. The number of quadratic terms in Eq. (10) severely impacts feasibility on NISQ machines with limited connectivity. Consequently, we need to understand the genesis of such contributions to the final formulation.

A quadratic contribution arises for each pair of variables that appears in at least one constraint. Of all constraints derived in Sec. 3.2, cardinality approximation in Eq. (7)—required for every join and every threshold value—contain the most variables (other constraints contain at most three binary variables, including slack variables). For Eq. (7), quadratic contributions arise for all variables tio_{tj}, tio_{tj} and pao_{tj} associated with join j. Pairs between these variables and variable cto_{rj}, as well as all binary variables needed to express the slack variable s_{rj}, are required. The number of threshold values has a large influence on quadratic contributions, as it impacts the number of required cardinality approximation constraints. The discretisation precision of continuous slack variables has a similarly large impact, as it influences the amount of binary slack variables in each of these constraints. This once more emphasises the need to carefully choose approximation precision in a quantum approach, which is unaccustomed from classical approaches.

3.5 Postprocessing

The final QUBO model in Eq. (1) can be used to solve JO using both, gate-based QAOA, and quantum annealing. In either case, we receive a sample set of possible solutions in the form of variable assignments together with the corresponding value of Eq. (1) that indicates the quality of a solution.

The result set must be mapped back to the initial JO problem. Since QPUs can deliver optimal, valid, and invalid solutions owing to hardware imperfections, we need to verify solution in a postprocessing step.

by satisfying $Sx = b$, where $S \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. A *valid* solution satisfies to constraints, but is not optimal.

Since non-optimal and invalid solutions are penalised, it is often possible to filter by the magnitude of the final penalty value. Caused by the multi-step reduction to QUBO, a large penalty indicates an invalid solution to the BILP problem with at least one constraint violation, but does not necessarily imply an invalid JO solution.

A solution is valid when an unambiguous, valid join tree can be derived from the assignment of QUBO variables, even if some constraints (*e.g.*, those relevant for the calculation of intermediate cardinalities) are violated. Instead of judging a solution by its penalty value, we consider the value assignments for all tii variables, which indicate the relations selected as the inner join operands, and verify whether each inner operand is uniquely represented by exactly one input relation. The final relation, representing the outer operand of the first join, is then given by process of elimination.

To judge solution quality, we calculate costs of the resulting join trees, and determine the best join order among all valid solutions.

4 ASSESSING STATE-OF-THE-ART QPUS

Current NISQ machines are not expected to solve practically relevant instances of JO. Nonetheless, an evaluation of our approach on actual hardware is important to gain insights on what limitations should be addressed best in future QPUs so that they can gain speedups over classical approaches. Note that we deliberately do not compare our results with existing classical approaches as these are of limited informative value, following recommendations of the quantum computing community [54] (it is even a hard problem to classically ascertain achieved quantum advantage [88]).

Our evaluation is based on QPUs offered by IBM and D-Wave. We focus on (a) maximal problem sizes (in terms of qubits, and circuit depth), (b) result quality, and (c) run/annealing times. All steps of our analysis are fully reproducible, and all QPU results are provided in a reproduction package.[5]

4.1 Experimental Setup

Experiments for the gate-based QAOA approach are executed on IBM Q Auckland (27 qubits, Falcon r5.11 topology) and Washington (127 qubits, Eagle r1 topology) systems. At the time of writing, Washington is the largest IBM Q system in terms of qubits, but has disadvantages in coherence time. Our experimental results confirm that for both, circuit transpilation actual execution, these disadvantages outweigh the size benefits.

For quantum annealing experiments, we resort to the D-Wave Advantage [53] system (5000 qubits, recently improved with a performance update, Pegasus topology).

Algorithmic Setup. Our approach, implemented in Python, prepares QUBO formulations for a given JO problem, and handles interaction with the QPUs via given cloud APIs. It relies on the `gurobipy` library [32] for formulating MILP and BILP problems. Packages `docplex` [38] (for IBM Q) and `qubovert` [67] (for Advantage), are used to formulate QUBO representations, which are then processed and passed to the QPU by IBM Qiskit [41] (IBM) and D-Wave Ocean [21] libraries.

[5] Available on Zenodo: https://doi.org/10.5281/zenodo.6508695; final version will contain a full reproduction package following [52].

The Qiskit QOAO library is used to generate quantum circuits that can be embedded onto IBM Q using the Qiskit transpiler (optimisation level 1). We run QAOA with $p = 1$, which creates one iteration of cost and mixing operators, since larger value of p lead to circuit depths beyond machine capability. Classical optimisation is performed with the Qiskit analytic quantum gradient descent optimiser (AQGD). We sample 1024 shots for each circuit executed on the QPU. Remaining parameters are set to their defaults.

For our experiments on the D-Wave Advantage system, we determine suitable embeddings using the heuristic minorminer tool provided in the D-Wave Ocean library [22]. We perform 1,000 annealing runs (matching the number of shots for IBM Q) for each JO problem. We experimentally determine suitable chain strengths, depending on the size of the respective JO problem. The exact chain strength values can be found in our provided reproduction package. We conduct experiments for varying annealing times, to study their impact on the solution quality. The remaining parameters are again set to their default values.

Queries. To ascertain a representative selection of larger queries with an increasing amount of relations and varying query graph types (chain, star and cycle queries), we rely on the method of Steinbrunn *et al.* [76], using the query generator code by Trummer [78].

Limitations of larger IBM Q machines, as explained above, necessitate to restrict experiments to 27 qubit (Auckland) machines. We can process basic queries that join at most three relations. Experiments do not reach practically relevant dimensions, but we provide qubit requirements for realistic problems on future QPUs below. Compared to IBM Q, the D-Wave Advantage system allows for embedding substantially larger queries.

Nonetheless, even with the restriction to three relations, we can generate different JO problems with varying properties. For instance, we generate problems that consider different numbers of predicates: For a query joining three relations, this provides us with four scenarios in total, where the number of predicates ranges between zero and three. As a result, for queries with less than two join predicates, cross products are required. For the remaining two scenarios with two and three predicates, we generate a chain query and a cycle query. These scenarios translate into varying qubit requirements, from 18 qubits for zero predicates up to 27 qubits for three predicates. This allows us to judge the impact of increasing problem dimensions, even with the restriction to three relations. Similarly, instead of increasing the number of predicates, we can vary the precision for discretising continuous variables, and generate problems of different dimensions, ranging from 18 to 27 qubits. This allows us to compare specific parameter settings.

Our goal is to find bounds on the dimension of JO problems for which QPUs determine viable solutions (since both, machines and algorithm, are stochastic, this is not guaranteed). QPU performance is influenced by a multitude of factors, algorithmic and physical, and deteriorates quickly for increasingly complex problems.

To focus on effects specific to the hardware instead of JO detail issues, we consider queries with integer logarithmic cardinalities and integer logarithmic predicate selectivities. This avoids discretisation issues with continuous variables. As discussed above, this has a large impact on the quadratic contributions to qubit scaling, and therefore the overall feasibility of our approach. Any upper

bounds derived in this scenario also apply to more general problems. Despite this relaxed scenario, our findings indicate much less optimistic predictions than published applications of QA to other query optimisation problems [79].

4.2 Experimental Results

We next discuss the results for our experimental evaluation of query optimisation on state-of-the-art QPUs. For both, IBM Q and D-Wave QPUs, we study the maximum problem dimensions for which we can determine a suitable embedding onto either architecture, as well as the performance of our approach on the actual QPUs.

4.2.1 *IBM Q Results.*

Transpilation. Figure 2 shows the distribution of circuit depths (based on 20 transpilations) necessary to embed QAOA circuits on two IBM Q devices using the heuristic Qiskit transpiler. As discussed, the circuit depth is crucial for the performance (and feasibility) of gate-based quantum computing, but varies intensely with the problem beyond size, rather different from classical approaches.

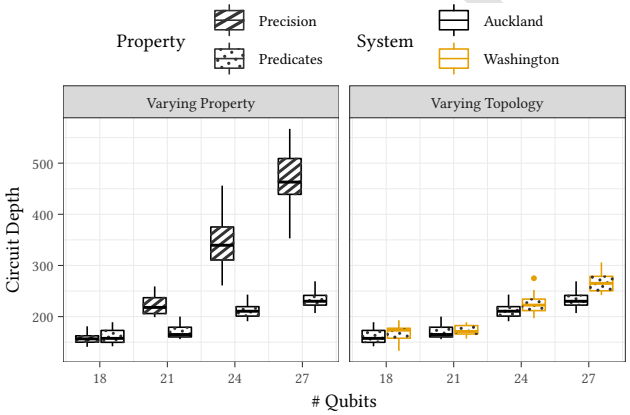

Figure 2: Circuit depths of various scenarios, IBM Q devices.

The left hand side of Fig. 2 compares circuit depths for problems with three relations and one threshold value, but different numbers of predicates and approximation precisions. By varying the approximation precision (striped boxplots) from zero to three decimal places (using zero predicates), we arrive at problems that can be mapped to 18, 21, 24 and 27 qubits. By varying the amount of predicates (dotted boxplots) from zero to three (using a discretisation precision of 0 decimal positions), we arrive at the same qubit requirements. Based on results of 20 Qiskit transpilation runs, we see that the increase in median circuit depth, but also in variance is considerably more pronounced for increasing precision than for increasing amounts of predicates. Equally interesting, the transpilation heuristic alone contributes substantial variance in the 24 and 27 qubit cases. Increasing the approximation precision therefore not only limits the feasible instance size (as fewer qubits are available for encoding relations or predicates), but also greatly impacts the circuit depth, thereby increasing the probability for gate errors and decoherence errors, and reducing result quality.

The right hand side of Fig. 2 compares embeddings of JO problems with an increasing number of predicates on IBM Q Auckland (27 qubits, Falcon r5.11 topology) and Washington (127 qubits, Eagle r1 topology) QPUs to analyse the impact of different qubit topologies on circuit depth. The increase is comparable to varying predicates on the left hand side, and we observe up to 70% difference depending on the slight variations of the topology for one single vendor (interestingly, the larger connectivity graph in terms of qubits leads to higher circuit depths).

We need to put these results in context by considering coherence times and average gate time g_{avg}—the cumulative gate times form a lax upper bound for practical utility, because for longer computation times, random results are to be expected. At the time of running the experiment,[6] the systems report coherence times of $T_1 = 151.13\mu s$, $T_2 = 138.72\mu s$ (Auckland), and $T_1 = 92.81\mu s$, $T_2 = 93.36\mu s$ (Washington). Average gate times are reported as 472.51ns (Auckland) and 550.41ns (Washington)—this, again, shows that a larger amount of qubits does not guarantee more favourable system-global properties.[7] Given these parameters, the approximate maximum depth d of a circuit to not exceed $T1$ or $T2$ is given by $d = \lfloor \min(\mathrm{T1}, \mathrm{T2}) / g_{\mathrm{avg}} \rfloor$.

QPU Performance. The previous results indicate that we cannot expect meaningful results for a high discretisation precision. Likewise, the required circuit depth rapidly exhaust coherence times of the Washington QPU. Experiments are therefore restricted to the 27-qubit Auckland QPU, with minimum precision, which again restricts us to problems with at most three relations.

Table 2: Solution quality for JO with three relations and varying amounts of predicates and QAOA iterations. Probabilities for finding *valid* and *optimal* solutions are based on 1024 shots on the IBM Q Auckland (27 qubits).

Predicates/Qubits		0/18	1/21	2/24	3/27
Valid solutions	20 Iter.	13%	11%	7%	13%
	50 Iter.	12%	8%	10%	13%
Optimal solutions	20 Iter.	4%	3%	2%	5%
	50 Iter.	3%	3%	5%	3%

Table 2 shows the QAOA results for 1,024 shots performed on the Auckland QPU, for queries with increasing predicate numbers, and provides the fraction of measurement shots that lead valid or optimal solutions. Note that for each query optimised on the QPU, the values of all solutions exceeded the minimal penalty, indicating at least one BILP constraint violation and once more demonstrating the imperfections of current QPUs. However, as discussed earlier, we consider each solution valid as long as it unambiguously corresponds to a valid join tree.

For all considered problems, optimal solutions are determined by the QPU. Interestingly, for increasing problem dimensions, we do not observe a consistent decrease in the ratios of either valid or optimal solutions. Likewise, increasing the number of QAOA iterations has no consistent impact on solution quality. Due to

[6] IBM QPUs undergo periodic recalibration operations to optimise T_1 and T_2, so the values are not stable across different experiments.

[7] *Quantum volume* [56] has been suggested as more balanced measure that weighs various characteristics of QPUs; interestingly, it is 64 for both QPUs in our experiments.

current QPU limitations, it remains an open question how the solution quality scales for larger search spaces (*i.e.*, more relations).

Finally, consider the time t_s required to perform the actual circuit sampling, and the overall QPU time t_{qpu} that includes additional initialisation steps and communication overhead (but does *not* take into account any delays spent in time-sharing queues that are necessary for shared cloud access to the machine). For 0 predicates, t_s = 77.9ms and t_{qpu} = 9.74s. The times intervals increase to t_s = 113.70ms and t_{qpu} = 10.35s for three predicates. The overall QPU time is *orders of magnitude* larger than the sampling time, and the problem size has *negligible* impact on the time it takes to receive sampled results for a QAOA circuit from the QPU. This must be taken into account when evaluating optimistic claims on possible QPU speedups based on solely t_s, as seen in the literature.

4.2.2 D-Wave Results. Our experimental analysis of the D-Wave Advantage system first determines upper bounds on the solvable problem dimensions by embedding the QUBO formulation onto the hardware graph, and then analyses the achievable result quality for given annealing times.

Embeddings. Fig. 3 depicts the embedding results for varying JO problems. We analyse the scaling of the number of physical qubits required for an embedding.

Firstly, consider scalability for an increasing number of relations for varying query graph types at minimum approximation precision. The type of query graph (chain, star, and cycle) has a negligible impact on the embeddings (top part). Cycle queries result in slightly larger embeddings compared to chain and star. For cycle queries, one additional predicate is required, which leads to a small overhead in QUBO variables.

For all three query graph types, we can determine embeddings for queries with up to 15 relations at minimum approximation precision, which provides an upper bound for solvable problem dimensions. The required physical qubits scale quadratically with the number of joined relations. As we show later, the number of binary variables (and therefore logical qubits) also scales quadratically with the number of relations. Embedding a JO problem onto the D-Wave topology therefore merely results in a linear qubit overhead.

However, increasing the approximation precision by more threshold values (bottom part) significantly increases the size of embeddings, particularly for smaller ω. For eight relations, we can include up to 20 threshold values at $\omega = 1$ (0 decimal positions), whereas merely six threshold values at $\omega = 0.01$ (two decimal positions), and three at $\omega = 0.0001$ (four decimal positions) are possible.

QPU Performance. Our embedding results provide an upper bound on the problem *size* that can be worked on the D-Wave Advantage system. To evaluate the practical feasibility of JO, we must also consider the achievable solution *quality*. Table 3 shows probabilities to reach valid and optimal solutions for queries with up to five relations (minimal discretisation precision, one threshold value).

Solution quality is not much influenced by query graph topology, but a steep decline in quality comes with increasing relations. In general, it suffices if one among all annealing shots corresponds to a (near-)optimal solution. For three and four relations, optimal solutions are determined sufficiently often. From five relations onward, typically *none* of the 1,000 shots include even a single valid solution, independent of query graph type and annealing time. While it is possible to embed larger problems onto the QPU, the approach stops being feasible for current D-Wave QPUs for more than four relations, which eliminates a potential advantage over

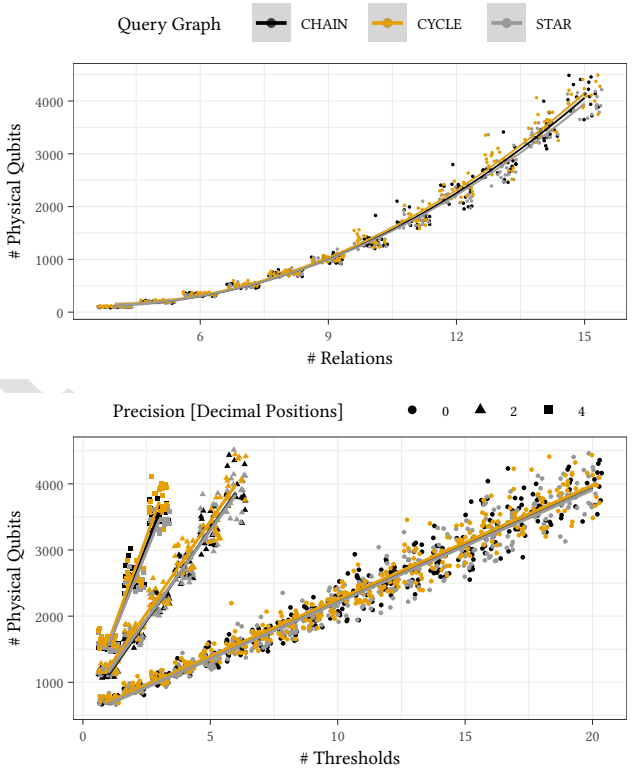

Figure 3: Physical qubits required to embed JO onto D-Wave Advantage. *Top*: Varying relations for random JO instances with $\omega = 0$. *Bottom*: Varying approximation precision for a fixed JO instance with eight relations.

Table 3: Average fraction of valid and optimal solutions obtained in 1,000 annealing runs over 20 JO experiments, depending on annealing time (Δt) and query graph type.

Query Graph	Δt [μs]	3 Relations Valid	3 Relations Opt.	4 Relations Valid	4 Relations Opt.	5 Relations Valid	5 Relations Opt.
Chain	20	30.0%	7.96%	1.53%	0.18%	0.07%	0%
	60	31.0%	8.68%	1.75%	0.17%	0.07%	0%
	100	33.0%	7.94%	1.86%	0.16%	0.05%	0%
Star	20	-	-	1.93%	0.31%	0.02%	0%
	60	-	-	2.10%	0.29%	0.02%	0%
	100	-	-	1.77%	0.17%	0.02%	0%
Cycle	20	25.49%	9.44%	3.10%	0.28%	0.02%	0%
	60	27.43%	10.26%	2.86%	0.29%	0.02%	0%
	100	27.22%	10.14%	3.20%	0.36%	0.03%	0%

gate-based QPUs. The impact of annealing times is minimal, which agrees with previous experience (see, *e.g.*, Refs. [46, 72]).

Our experimental analysis clearly shows that contemporary QPUs are not practically useful to solve JO problems. Limited amounts of qubits, limited connectivity, and lack of robustness against noise reduce capabilities and performance. These issues will eventually (likely in the very long term [44]) be addressed by error-corrected, perfect QPUs. To progress towards earlier QPU utility, we now pursue two strands: First, we derive an upper bound on the amount of required qubits for JO; this allows us to predict how QPU capacity will need to grow to accommodate realistic problem sizes. Then, we address the question of which properties of QPUs should be improved to progress towards practical utility.

5 FORMAL ANALYSIS

We now derive upper bounds on the amount of logical qubits based on the BILP model, where each variable corresponds to one qubit.

Slack variables, as required to handle inequalities, contribute substantially to the overall number of qubits. Before we can derive an upper bound for the number of *binary* slack variables, we need to bound the value of a continuous slack variable:

Lemma 5.1. *The value for a continuous slack variable s_{rj} is bounded by $s_{rj} \leq c_{j_{max}}$, where $c_{j_{max}}$ denotes the maximum logarithmic cardinality of the outer operand of join j.*

Proof. By Eq. (8), an upper bound for s_{rj} is given by

$$\log(\theta_r) + \infty_{rj} \geq \log(\theta_r) + cto_{rj}\infty_{rj} - c_j = s_{rj}. \tag{11}$$

Since our model assumes uncorrelated predicates, the fraction of surviving tuples after joining multiple relations is given by the product of selectivities of all applicable predicates. Since $Sel(p) > 0$, an intermediate result contains at least one tuple, hence $c_j \geq 0$. The upper bound depends on the constant ∞_{rj}, which needs to be sufficiently large to satisfy the constraint by activating cto_{rj}, but can otherwise be freely chosen.

Since we seek the smallest upper bound for s_{rj}, we next specify a lower bound for ∞_{rj}. Following Eq. (7), this lower bound is given by $\infty_{rj} \geq c_{j_{\max}} - \log(\theta_r) \geq c_j - \log(\theta_r)$. Setting ∞_{rj} to its lower bound, and inserting it into Eq. (11), produces $s_{rj} \leq c_{j_{\max}}$. □

Lemma 5.2. *The maximum logarithmic cardinality $c_{j_{max}}$ for the outer operand of join j is given by $c_{j_{max}} = \sum_{t=0}^{j+1} \log(Card(t))$, where $Card(k) \geq Card(l) \forall k < l$.*

Proof. The logarithmic cardinality c_j for join j is given by $c_j = \sum_t \log(Card(t)) tio_{tj} + \sum_p \log(Sel(p)) pao_{pj}$. The cardinality may be reduced by applying predicates, since $0 < Sel(p) \leq 1$, making the logarithmic values negative. Since we consider the maximum cardinality, we set all variables $pao_{pj} = 0$, disregarding any predicates. The outer operand of join j contains exactly $j+1$ relations. The logarithmic intermediate cardinality is then maximised if the outer operand for join j contains the first $j+1$ relations out of a list of relations sorted in descending order by cardinalities. □

Theorem 5.3. *Given T relations, J joins, P predicates and R threshold values, an upper bound for the number of variables is given by $n \leq 2TJ + (3P+R)(J-1) + T + R\sum_{j=1}^{J-1}\left(\left\lfloor \log_2\left(\frac{c_j}{\omega}\right)\right\rfloor + 1\right)$, where ω is the approximation precision for the continuous slack variables.*

Proof. The number of binary variables is $n_{\text{pec}} + n_{\text{sl}}$, where n_{pec} is the number of problem-encoding variables given in Sec. 3.2, and n_{sl} counts slack variables for equality conversion.

First, we specify an upper bound for n_{pec}. As explained in Sec. 3.2, variables tii_{tj} and tio_{tj} are added $T \cdot J$ times, whereas variables pao_{pj} and cto_{rj} are added $J-1$ times for P predicates and R threshold values. Depending on the concrete JO problem, we may be allowed to prune variables. A variable cto_{rj} is unnecessary if the logarithmic cardinality of the outer operand for join j can never exceed the logarithmic threshold value $\log(\theta_r)$. We therefore prune every variable cto_{rj} if $c_{j_{\max}} \leq \log(\theta_r)$. The number of variables when no pruning is possible then gives the upper bound $n_{\text{pec}} \leq 2TJ + (P+R)(J-1)$. To specify an upper bound for n_{sl}, consider that one binary slack variable is needed for each inequality constraint expressed by Eqns (4),(5). As such, $T + 2P(J-1)$ variables are required in these cases. In turn, multiple binary slack variables are needed to approximate continuous slack variables for constraints expressed by Eq. (7). Following from Lemma 5.1 and Eq. (9), an upper bound for the number of binary slack variables n_b required to discretise all continuous slack variables for $J-1$ joins and R threshold variables is given by $n_b \leq R\sum_{j=1}^{J-1}\left(\lfloor \log_2\left(c_{j_{\max}}/\omega\right)\rfloor + 1\right)$. Note that we specify an upper bound, since a constraint is only required if the corresponding variable cto_{rj} has not been pruned. Considering the other inequality constraints, the upper bound for n_{sl} is given by $n_{\text{sl}} \leq T + 2P(J-1) + n_b$, which leads to the upper bound for the overall number of binary variables $n = n_{\text{pec}} + n_{\text{sl}}$ as

$$n \leq 2TJ + (3P+R)(J-1) + T + R\sum_{j=1}^{J-1}\left(\left\lfloor \log_2\left(\frac{c_{j_{\max}}}{\omega}\right)\right\rfloor + 1\right). \qquad \square$$

6 DB-QPU CO-DESIGN FOR JOIN ORDERING

We now commence to deriving recommendations for future QPU designs customized towards serving as co-processors in databases.

6.1 Logical Qubit Scaling

Fig. 4 visualises upper bounds for a variety of JO problems with up to 64 relations and different discretisation precisions. We measure for cyclic query graphs, as they need one additional join predicate compared to other types, and require the largest number of qubits.

The upper bound for the required logical qubits scales quadratically with the number of relations, the dominating scaling factor. Increase in discretisation precision has comparatively little impact on the upper bound compared to the number of relations, even if the difference in terms of qubits can reach more than 50% in some scenarios (top right). Nonetheless, as we have shown in the experimental analysis, this seemingly minor influence can have a decisive impact on the feasibility on current QPUs.

The required amount of qubits rapidly exceeds the capacity of current gate-based NISQ machines. Solving the largest problems considered with classical MILP solvers [80], where queries with 60 relations are joined, requires a QPU with more than 20,000 qubits, which is out of reach for the foreseeable future. However, a QPU offering 1,000 logical qubits can, depending on the approximation

and discretisation precision, solve problems with up to 13 relations, and therefore optimise queries roughly equal in size to those considered in the JO benchmark by Leis et al. [47]. Vendor roadmaps claim to offer such QPUs within the next few years [37].

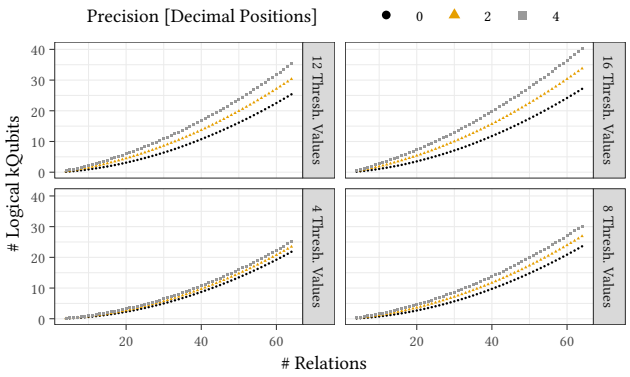

Figure 4: Upper bounds for logical qubits for JO with varying approximation (thresh. values) and discretisation precisions.

6.2 QPU Extrapolation

Future QPUs need to improve gate errors and decoherence. Their impact amplifies with increasing quantum circuit depth, and we have observed that even minor alterations to the topology lead to substantial variations in circuit depth. We now analyse the feasibility of solving JO problems on hypothetical improved QPU topologies.

Scenarios. We extrapolate new topologies in three ways: By increasing the amount of qubits based on a structural extension of the available connectivity graph, by augmenting the graph with additional connections, and by using different quantum gate sets. To quantify possible improvements, we again solve JO on queries generated by the code of Ref. [78]. We consider problems with two threshold values, and minimal discretisation precision (*i.e.*, $\omega = 1$).

Size Extrapolation. We consider baseline designs from IBM [39] (127 qubit Washington), IonQ [40], and Rigetti [69] (80-qubit Aspen-M), that is, their topologies and native gate sets. Similarly to IBM Q, Rigetti QPUs are based on superconducting qubits, whereas IonQ QPUs are based on trapped ions. QPUs based on this physical principle are more stable and offer full connectivity between qubits, whereas QPUs featuring superconducting qubits offer less qubit connectivity and stability, but feature faster gates [9, 60].

IBM and Rigetti topologies are based on repeating patterns, allowing for straightforward extrapolation to larger numbers of qubits (the detailed logic is given in the reproduction package). IonQ QPUs do not require extrapolation, as their topology is a complete mesh.

Density Extrapolation. We augment existing topologies by adding new connections between previously non-adjacent qubits. A fully connected topology with n qubits contains $N = n(n-1)/2$ edges. When the baseline topology includes M edges, we quantify the *extended connectivity* as $d = m/(N-M)$, where m denotes the amount of added notes. d lies in the interval $[0, 1]$, and interpolates between the baseline topology ($d = 0$) and a complete mesh ($d = 1$). We assume connections between non-adjacent qubits with close topological proximity are more likely to be featured in future QPUs than between far-distant qubits. Instead of uniformly sampling from the set of all missing connections, we favour extending the connectivity between non-adjacent qubits with close proximity.

Consider the set of connections C_δ between qubits with distance δ. Starting with $\delta = 2$, we uniformly add connections sampled from C_δ, until the desired density is reached, or until all elements of C_δ have been added, in which case we restart from $C_{\delta+1}$.

Results. Figure 5 shows the depths of the quantum circuits for combinations of randomly generated join ordering problems (as before), QPU architectures (varying topology and gate set), and transpilation methods. Concerning gate sets, we study the impact of transpilation onto the *native* set, which involves replacing any unsupported gate operation with an effectively equivalent chain of native operations, versus *unrestricted* gate sets, where we assume the QPU to natively support any possible gate operation. For density 0 (*i.e.*, baseline topology), we notice a substantial increase in circuit depth for an increasing number of relations (notice the graph is in log scale!) that quickly exceeds NISQ capabilities. However, even very moderately increased densities (0.05 to 0.1) lead do much smaller circuit depths (up to one order of magnitude for the native gate sets). On IBM Q, relative differences in circuit depth are about identical between density 0 and 0.05, and 0.75 and 1–albeit a fully meshed qubit network, as described by density 1, is obviously impossible to achieve in a two-dimensional structure. This indicates that adding even a small amount of extra connections can substantially impact the utility of QPUs for join ordering problems. Similar observations, albeit not as pronounced, hold for Rigetti.

Transpiling a circuit onto the native gate sets (compared to unrestricted gates) also significantly increases depth as compared to a (hypothetical) unrestricted gate set for the Rigetti QPU, but does not significantly impact IBM Q. Nonetheless, judiciously expanding the set of supported native operations may also enhance problem feasibility of future QPUs as an alternative to improved topologies.

Circuit synthesis is a widely investigated topic in the classical domain, and has gained attention for QPUs (see, *e.g.*, Refs. [4, 15, 82]). Recent work addresses, for instance, noise reduction in QPUs [84], or how to approximate quantum circuits [83]. Our experiments therefore consider two (sufficiently mature) transpilation approaches, Qiskit [41] and tket [75]. While the scaling behaviour is essentially identical for increased connectivity density, we observe an overhead of typically 100% for tket over Qiskit for the superconducting platforms. Since both transpilers produce comparable results for complete meshes as in the IonQ case, we conclude that their capabilities of the approaches are similar for boundary cases, but need to be carefully evaluated in all other cases.

The IonQ platform features full connectivity between qubits as baseline, and is therefore not subjected to experiments with increasing density. The resulting circuit depths seem ideal compared to the superconducting platforms. Yet, the amount of qubits that can be supported by this physical technology is limited by the amount of individual ions that can be caught in a trap; current technology allows for tens of ions [28], and major improvements are not to be expected. Considering the predictions of Fig. 4, the advantages in circuit depth are therefore compensated by the lack of qubits.

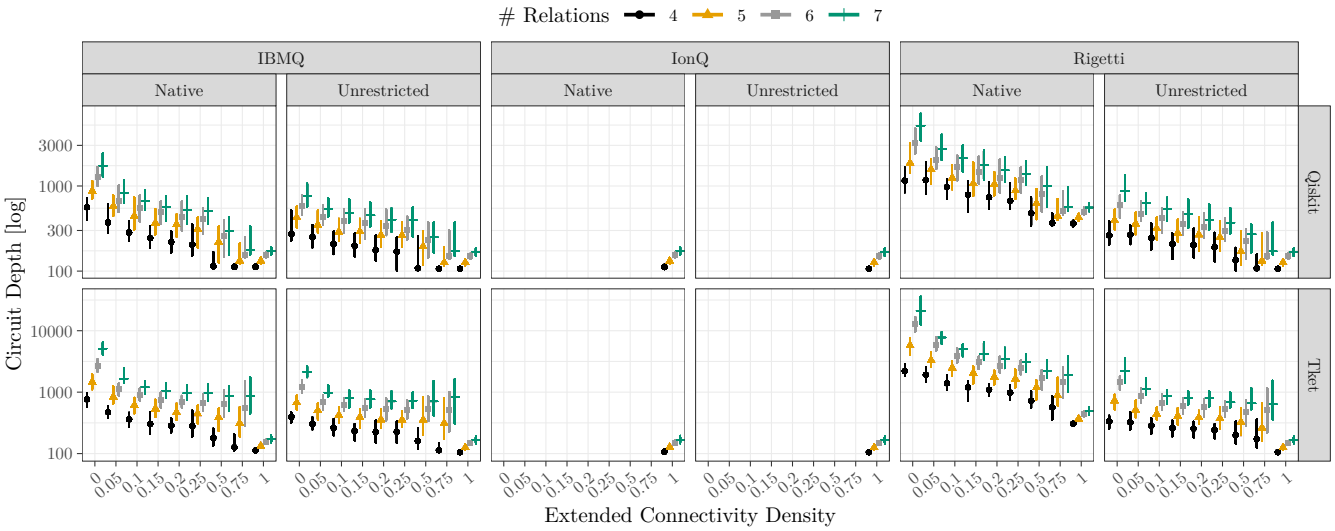

Figure 5: Circuit Depths for hypothetical future QPUs on varying join order instances. Experiments are performed with different query graph topologies, but these are not visualised since no relevant differences arise.

7 RELATED WORK

Join ordering is among the most studied problems in database research [29, 45, 47, 55, 58, 59, 76, 80, 86]. Yet, no published work exists for JO with QCs to the best of our knowledge. Very few endeavours address the more general use of QCs in databases. This is in stark contrast to the growing interest in quantum computing in other fields [17, 18, 36, 42, 43, 62, 63, 66, 70, 71, 81, 85].

The multi-query optimisation problem (MQO) was analysed by Trummer and Koch [79] for quantum annealing at the VLDB conference. They experimentally compare QA performance with classical approaches. Due to the hardware limitations, they had to focus on small-scale problems and could only speculate on speedups for this specific class. Frankhauser *et al.* [23] addresses MQO for gate-based QPUs. DB transaction scheduling, was studied by Groppe and Groppe [30], and Bittner and Groppe [11, 12].

Related work on quantum computing (*e.g.*, Refs. [9, 60]), annealing (*e.g.*, Refs [3, 53]), and the utilised algorithms like QAOA [7, 24, 25, 33–35] has been intensively discussed in Sec. 2.

8 DISCUSSION AND CONCLUSION

Quantum computing is a new paradigm that promises—grounded on theoretical insights and guarantees [9], but also based on first experimental results on certain problems [6]—speedups for computational problems over classical approaches. The technology is structurally apt for many aspects of database systems, including query optimisation problems. So far, the use of QCs for databases is extremely underexplored. Yet, our work leaves a Janus-faced impression: On the one hand, we show that *current* NISQ-systems are far away from producing benefits, or even from handling realistically sized instances, and the classical operations required to access them can eliminate any quantum advantage. This, at least, paints a more sober picture than initial optimistic evaluations of the technology. On the other hand, we show that with relatively minor adaptations, QPU-DB performance can be substantially enhanced. This prompts to use co-design approaches to create QPU-DB accelerators, which is reasonable given that databases are among the commercially most important applications of computer science.

QPUs are typically accessed via cloud services, but we envision their use as *local* co-processors to accelerate query processing. This avoids the impact of network latencies (that govern QPU cloud services), which can easily eliminate quantum speedups.

Our results and predictions show that a multitude of factors influence the performance of QC approaches on DB problems, ranging from unusual problem formulations in QUBO form that massively diverge from traditional implementation techniques, to unfavourable scaling caused by subtle issues like discretisation precision, to a complex interplay of physical implementation properties. This makes it impossible to delay QC integration into databases until sufficiently evolved hardware is available, but prompts the co-design of database accelerators. Simulating perfect quantum computers entails solving NP-hard problems, and adding the effects of noise and imperfections causes additional complexities. Consequently, we argue that co-design efforts should be based on step-wise empirical refinement leveraging expert knowledge from quantum computing, databases, and systems engineering.

Our work lays the ground for this endeavour: Our QUBO formulation of JO enables the experimental exploration of two major classes of QCs, and we identify relevant factors that inhibit scalability and practical utility. We also provide directions on designing QPUs favourable for join ordering problems.

Nonetheless, many open research problems remain, from efficient circuit generation that respects the influence of noise, to more targeted extensions of topologies that transcend our semi-stochastic approach, to considering alternative information encoding schemes that might (*e.g.*, by representing rational values in amplitudes instead of storing them in qubits) alleviate discretisation problems.

Yet such approaches are still challenged by major physical and algorithmic obstacles, and require intensive future interdisciplinary, focused research.

REFERENCES

[1] Tobias Achterberg, Robert E. Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. 2020. Presolve Reductions in Mixed Integer Programming. *INFORMS Journal on Computing* 32, 2 (2020), 473–506. https://doi.org/10.1287/ijoc.2018.0857

[2] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. 2008. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Rev.* 50, 4 (2008), 755–787.

[3] Tameem Albash and Daniel A. Lidar. 2018. Adiabatic quantum computation. *Rev. Mod. Phys.* 90 (Jan 2018), 015002. Issue 1. https://doi.org/10.1103/RevModPhys.90.015002

[4] Carmen G. Almudéver, Lingling Lao, Robert Wille, and Gian Giacomo Guerreschi. 2020. Realizing Quantum Algorithms on Real Quantum Computing Devices. In *2020 Design, Automation & Test in Europe Conference & Exhibition, DATE 2020, Grenoble, France, March 9-13, 2020*. IEEE, 864–872. https://doi.org/10.23919/DATE48585.2020.9116240

[5] S. Arora and B. Barak. 2006. *Computational Complexity: A Modern Approach*. Cambridge University Press. https://theory.cs.princeton.edu/complexity/book.pdf

[6] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (01 Oct 2019), 505–510. https://doi.org/10.1038/s41586-019-1666-5

[7] Boaz Barak and Kunal Marwaha. 2022. Classical Algorithms and Quantum Limitations for Maximum Cut on High-Girth Graphs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPICS.ITCS.2022.14

[8] Ethan Bernstein and Umesh Vazirani. 1997. Quantum Complexity Theory. *SIAM J. Comput.* 26, 5 (1997), 1411–1473. https://doi.org/10.1137/S0097539796300921 arXiv:https://doi.org/10.1137/S0097539796300921

[9] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S. Kottmann, Tim Menke, Wai-Keong Mok, Sukin Sim, Leong-Chuan Kwek, and Alán Aspuru-Guzik. 2022. Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.* 94 (Feb 2022), 015004. Issue 1. https://doi.org/10.1103/RevModPhys.94.015004

[10] Zhengbing Bian, Fabián Chudak, William Macready, and Geordie Rose. 2010. *The Ising model: Teaching an old problem new tricks*. Technical Report. D-Wave Systems Inc.

[11] Tim Bittner and Sven Groppe. 2020. Avoiding Blocking by Scheduling Transactions Using Quantum Annealing. In *Proceedings of the 24th Symposium on International Database Engineering & Applications* (Seoul, Republic of Korea) *(IDEAS '20)*. Association for Computing Machinery, New York, NY, USA, Article 21, 10 pages. https://doi.org/10.1145/3410566.3410593

[12] Tim Bittner and Sven Groppe. 2020. Hardware Accelerating the Optimization of Transaction Schedules via Quantum Annealing by Avoiding Blocking. *Open Journal of Cloud Computing (OJCC)* 7, 1 (2020), 1–21.

[13] P. I. Bunyk, Emile M. Hoskinson, Mark W. Johnson, Elena Tolkacheva, Fabio Altomare, Andrew J. Berkley, Richard Harris, Jeremy P. Hilton, Trevor Lanting, Anthony J. Przybysz, and Jed Whittaker. 2014. Architectural considerations in the design of a superconducting quantum annealing processor. *IEEE Transactions on Applied Superconductivity* 24, 4 (April 2014), 1–10.

[14] Lukas Burgholzer, Rudy Raymond, and Robert Wille. 2020. Verifying results of the IBM Qiskit quantum circuit compilation flow. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, Denver, CO, USA, 356–365.

[15] Lukas Burgholzer, Sarah Schneider, and Robert Wille. 2022. Limiting the Search Space in Optimal Quantum Circuit Mapping. In *27th Asia and South Pacific Design Automation Conference, ASP-DAC 2022, Taipei, Taiwan, January 17-20, 2022*. 466–471. https://doi.org/10.1109/ASP-DAC52403.2022.9712555

[16] Cristian S. Calude and Michael J. Dinneen. 2017. Solving the broadcast time problem using a D-Wave quantum computer. In *Advances in Unconventional Computing: Volume 1: Theory*. Springer International Publishing, Cham, 439–453.

[17] Yudong Cao, Shuxian Jiang, Debbie Perouli, and Sabre Kais. 2016. Solving Set Cover with Pairs Problem Using Quantum Annealing. *Scientific Reports* 6 (08 2016). https://doi.org/10.1038/srep33957

[18] Guillaume Chapuis, Hristo Djidjev, Georg Hahn, and Guillaume Rizk. 2019. Finding Maximum Cliques on a Quantum Annealer. *Journal of Signal Processing Systems* 91 (03 2019). https://doi.org/10.1007/s11265-018-1357-8

[19] Sophie Cluet and Guido Moerkotte. 1995. On the complexity of generating optimal left-deep processing trees with cross products. In *Database Theory — ICDT '95*. Springer Berlin Heidelberg, Berlin, Heidelberg, 54–67.

[20] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. 2014. *Integer programming*. Graduate Texts in Mathematics, Vol. 271. Springer International Publishing, Cham.

[21] D-Wave Systems Inc. 2022. Documentation for the Ocean SDK for solving problems on D-Wave quantum computers. https://docs.ocean.dwavesys.com/en/stable/

[22] D-Wave Systems Inc. 2022. Minorminer library for embedding Ising problems onto quantum annealers. https://docs.ocean.dwavesys.com/en/stable/docs_minorminer/source/intro.html

[23] Tobias Fankhauser, Marc E. Solèr, Rudolf M. Füchslin, and Kurt Stockinger. 2021. Multiple query optimization using a hybrid approach of classical and quantum computing. (July 2021). arXiv:2107.10508

[24] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. (Nov. 2014). arXiv:1411.4028

[25] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem. (Dec. 2014). arXiv:1412.6062

[26] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. 2000. Quantum computation by adiabatic evolution. (Jan. 2000). arXiv:quant-ph/0001106

[27] Edward Farhi and Aram W Harrow. 2016. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. https://doi.org/10.48550/arxiv.1602.07674 arXiv:1602.07674

[28] Nicolai Friis, Oliver Marty, Christine Maier, Cornelius Hempel, Milan Holzäpfel, Petar Jurcevic, Martin B. Plenio, Marcus Huber, Christian Roos, Rainer Blatt, and Ben Lanyon. 2018. Observation of Entangled States of a Fully Controlled 20-Qubit System. *Phys. Rev. X* 8 (Apr 2018), 021012. Issue 2. https://doi.org/10.1103/PhysRevX.8.021012

[29] Frederico A.C.A. Gonçalves, Frederico G. Guimarães, and Marcone J.F. Souza. 2014. Query join ordering optimization with evolutionary multi-agent systems. *Expert Systems with Applications* 41, 15 (2014), 6934–6944. https://doi.org/10.1016/j.eswa.2014.05.005

[30] Sven Groppe and Jinghua Groppe. 2021. Optimizing Transaction Schedules on Universal Quantum Computers via Code Generation for Grover's Search Algorithm. In *25th International Database Engineering & Applications Symposium* (Montreal, QC, Canada) *(IDEAS 2021)*. Association for Computing Machinery, New York, NY, USA, 149–156. https://doi.org/10.1145/3472163.3472164

[31] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*. ACM, New York, New York, USA, 212–219.

[32] Gurobi Optimization, LLC. 2022. Gurobi optimizer reference manual. https://www.gurobi.com

[33] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G. Rieffel, Davide Venturelli, and Rupak Biswas. 2019. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms* 12, 2 (2019). https://doi.org/10.3390/a12020034

[34] Stuart Hadfield, Zhihui Wang, Eleanor G. Rieffel, Bryan O'Gorman, Davide Venturelli, and Rupak Biswas. 2017. Quantum Approximate Optimization with Hard and Soft Constraints. In *Proceedings of the Second International Workshop on Post Moores Era Supercomputing* (Denver, CO, USA) *(PMES'17)*. Association for Computing Machinery, New York, NY, USA, 15–21. https://doi.org/10.1145/3149526.3149530

[35] David Headley, Thorge Müller, Ana Martin, Enrique Solano, Mikel Sanz, and Frank K. Wilhelm. 2020. Approximating the Quantum Approximate Optimisation Algorithm. arXiv:2002.12215 https://arxiv.org/abs/2002.12215

[36] Hristo N. Djidjev, Guillaume Chapuis, Georg Hahn, and Guillaume Rizk. 2018. Efficient Combinatorial Optimization Using Quantum Annealing.

[37] IBM. 2021. IBM's roadmap for building an open quantum software ecosystem. https://research.ibm.com/blog/quantum-development-roadmap

[38] IBM. 2022. IBM Decision Optimization CPLEX Modeling for Python. https://ibmdecisionoptimization.github.io/docplex-doc/

[39] IBM Quantum. 2022. Cloud access to quantum computers provided by IBM. https://quantum-computing.ibm.com

[40] IBM Quantum. 2022. IonQ technology. https://ionq.com/technology

[41] IBM Quantum. 2022. Qiskit: An Open-source Framework for Quantum Computing. https://qiskit.org/

[42] K. Ikeda, Y. Nakamura, and T. S. Humble. 2019. Application of Quantum Annealing to Nurse Scheduling Problem. *Sci Rep* (2019). https://doi.org/10.1038/s41598-019-49172-3

[43] Hirotaka Irie, Goragot Wongpaisarnsin, Masayoshi Terabe, Akira Miki, and Shinichirou Taguchi. 2019. Quantum Annealing of Vehicle Routing Problem

with Time, State and Capacity. In *Quantum Technology and Optimization Problems*, Sebastian Feld and Claudia Linnhoff-Popien (Eds.). Springer International Publishing, Cham, 145–156.

[44] Morten Kjaergaard, Mollie E. Schwartz, Jochen Braumüller, Philip Krantz, Joel I.-J. Wang, Simon Gustavsson, and William D. Oliver. 2020. Superconducting Qubits: Current State of Play. *Annual Review of Condensed Matter Physics* 11, 1 (2020), 369–395. https://doi.org/10.1146/annurev-conmatphys-031119-050605

[45] Ilya Kolchinsky and Assaf Schuster. 2018. Join Query Optimization Techniques for Complex Event Processing Applications. (2018). arXiv:1801.09413

[46] Tom Krüger and Wolfgang Mauerer. 2020. 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*. 445–450. https://doi.org/10.1145/3387940.3391472

[47] Viktor Leis, Bernhard Radke, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2018. Query optimization through the looking glass, and what we found running the join order benchmark. *The VLDB Journal* 27 (2018), 643–668.

[48] Mark Lewis and Fred Glover. 2017. Quadratic Unconstrained Binary Optimization Problem Preprocessing: Theory and Empirical Analysis. (May 2017). arXiv:1705.09844

[49] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. 2021. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics* 17, 9 (01 Sep 2021), 1013–1017. https://doi.org/10.1038/s41567-021-01287-z

[50] Andrew Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2 (2014), 5.

[51] Jirí Matousek and Robin Thomas. 1992. On the complexity of finding iso- and other morphisms for partial k-trees. *Discret. Math.* 108 (1992), 343–364.

[52] Wolfgang Mauerer and Stefanie Scherzinger. 2022. 1-2-3 Reproducibility for Quantum Software Experiments. arXiv:2201.12031 [cs.SE] https://arxiv.org/abs/2201.12031

[53] Catherine McGeoch and Pau Farré. 2020. *The D-Wave Advantage system: An overview*. Technical Report 14-1049A-A. D-Wave Systems Inc.

[54] Catherine C. McGeoch. 2019. Principles and Guidelines for Quantum Performance Analysis. In *Quantum Technology and Optimization Problems*. Springer International Publishing, Cham, 36–48.

[55] Guido Moerkotte. 2020. Building query compilers. https://pi3.informatik.uni-mannheim.de/~moer/querycompiler.pdf

[56] Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, Jerry M Chow, Andrew Cross, Daniel J Egger, Stefan Filipp, Andreas Fuhrer, Jay M Gambetta, Marc Ganzhorn, Abhinav Kandala, Antonio Mezzacapo, Peter Müller, Walter Riess, Gian Salis, John Smolin, Ivano Tavernelli, and Kristan Temme. 2018. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology* 3, 3 (jun 2018), 030503. https://doi.org/10.1088/2058-9565/aab822

[57] Prakash Murali, David C. Mckay, Margaret Martonosi, and Ali Javadi-Abhari. 2020. Software mitigation of crosstalk on noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, New York, NY, USA, 1001–1016.

[58] Thomas Neumann. 2009. Query Simplification: Graceful Degradation for Join-Order Optimization. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data* (Providence, Rhode Island, USA) *(SIGMOD '09)*. Association for Computing Machinery, New York, NY, USA, 403–414. https://doi.org/10.1145/1559845.1559889

[59] Thomas Neumann and Bernhard Radke. 2018. Adaptive Optimization of Very Large Join Queries. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 677–692. https://doi.org/10.1145/3183713.3183733

[60] Michael A. Nielsen, Isaac Chuang, and Lov K. Grover. 2002. Quantum computation and quantum information. *American Journal of Physics* 70, 5 (April 2002), 558–559.

[61] B. O'Gorman, R. Babbush, A. Perdomo-Ortiz, A. Aspuru-Guzik, and V. Smelyanskiy. 2015. Bayesian network structure learning using quantum annealing. *The European Physical Journal Special Topics* 224, 1 (2015), 163–188.

[62] Elijah Pelofske, Georg Hahn, and Hristo Djidjev. 2019. Solving large minimum vertex cover problems on a quantum annealer. 76–84. https://doi.org/10.1145/3310273.3321562

[63] W. Peng, B. Wang, and F. et al. Hu. 2019. Factoring larger integers with fewer qubits via quantum annealing with optimized parameters. *Sci. China Phys. Mech. Astron.* (2019). https://doi.org/10.1007/s11433-018-9307-1

[64] Asher Peres and Leslie E. Ballentine. 1995. Quantum Theory: Concepts and Methods. *American Journal of Physics* 63, 3 (1995), 285–286. https://doi.org/10.1119/1.17946

[65] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (2018), 79.

[66] Quantum Technology and Application Consortium - QUTAC, Bayerstadler, Andreas, Becquin, Guillaume, Binder, Julia, Botter, Thierry, Ehm, Hans, Ehmer, Thomas, Erdmann, Marvin, Gaus, Norbert, Harbach, Philipp, Hess, Maximilian, Klepsch, Johannes, Leib, Martin, Luber, Sebastian, Luckow, Andre, Mansky, Maximilian, Mauerer, Wolfgang, Neukart, Florian, Niedermeier, Christoph, Palackal, Lilly, Pfeiffer, Ruben, Polenz, Carsten, Sepulveda, Johanna, Sievers, Tammo, Standen, Brian, Streif, Michael, Strohm, Thomas, Utschig-Utschig, Clemens, Volz, Daniel, Weiss, Horst, and Winter, Fabian. 2021. Industry quantum computing applications. *EPJ Quantum Technol.* 8, 1 (2021), 25. https://doi.org/10.1140/epjqt/s40507-021-00114-x

[67] qubovert. 2022. The one-stop package for formulating, simulating, and solving problems in boolean and spin form. https://qubovert.readthedocs.io/en/latest/index.html

[68] Eleanor Rieffel and Wolfgang Polak. 2011. *Quantum computing: A gentle introduction*. MIT Press, Cambridge, MA.

[69] Rigetti Computing. 2022. Rigetti quantum processors. https://qcs.rigetti.com/qpus

[70] S. Feld, M. Friedrich, and C. Linnhoff-Popien. 2018. Optimizing Geometry Compression Using Quantum Annealing. In *2018 IEEE Globecom Workshops (GC Wkshps)*. 1–6. https://doi.org/10.1109/GLOCOMW.2018.8644358

[71] S. Yarkoni, A. Plaat, and T. Back. 2018. First Results Solving Arbitrarily Structured Maximum Independent Set Problems Using Quantum Annealing. In *2018 IEEE Congress on Evolutionary Computation (CEC)*. 1–6. https://doi.org/10.1109/CEC.2018.8477865

[72] Irmi Sax, Sebastian Feld, Sebastian Zielinski, Thomas Gabor, Claudia Linnhoff-Popien, and Wolfgang Mauerer. 2020. Approximate Approximation on a Quantum Annealer. In *Proceedings of the 17th ACM International Conference on Computing Frontiers* (Catania, Sicily, Italy) *(CF '20)*. Association for Computing Machinery, New York, NY, USA, 108–117. https://doi.org/10.1145/3387902.3392635

[73] Manuel Schönberger, Maja Franz, Stefanie Scherzinger, and Wolfgang Mauerer. 2022. Peel | pile? Cross-framework portability of quantum software. In *19th IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE, Honolulu, HI, USA.

[74] P. W. Shor. 1994. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE Comput. Soc. Press, Santa Fe, NM, USA, 124–134.

[75] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. 2020. t|ket⟩: a retargetable compiler for NISQ devices. *Quantum Science and Technology* 6, 1 (nov 2020), 014003. https://doi.org/10.1088/2058-9565/ab8e92

[76] Michael Steinbrunn, Guido Moerkotte, and Alfons Kemper. 1997. Heuristic and randomized optimization for the join ordering problem. *The VLDB journal* 6 (1997), 191–208.

[77] Swamit S. Tannu and Moinuddin K. Qureshi. 2019. Not All Qubits Are Created Equal: A Case for Variability-Aware Policies for NISQ-Era Quantum Computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, New York, NY, USA, 987–999.

[78] Immanuel Trummer. 2016. Query Optimizer Library. https://github.com/itrummer/query-optimizer-lib

[79] Immanuel Trummer and Christoph Koch. 2016. Multiple query optimization on the D-Wave 2X adiabatic quantum computer. *Proceedings of the VLDB Endowment* 9, 9 (May 2016), 648–659.

[80] Immanuel Trummer and Christoph Koch. 2017. Solving the join ordering problem via mixed integer linear programming. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, New York, NY, USA, 1025–1040.

[81] D. Venturelli and A.. Kondratyev. 2019. Reverse quantum annealing approach to portfolio optimization problems. *Quantum Mach. Intell.* (2019), 17–30. https://doi.org/10.1007/s42484-019-00001-w

[82] Robert Wille and Rolf Drechsler. 2022. Introduction to the Special Issue on Design Automation for Quantum Computing. *ACM J. Emerg. Technol. Comput. Syst.* 18, 1 (2022), 10:1–10:2. https://doi.org/10.1145/3485041

[83] Ellis Wilson, Frank Mueller, Lindsay Bassman, and Costin Iancu. 2021. Empirical evaluation of circuit approximations on noisy quantum devices. In *SC '21: The International Conference for High Performance Computing, Networking, Storage and Analysis, St. Louis, Missouri, USA, November 14 - 19, 2021*. 96:1–96:15. https://doi.org/10.1145/3458817.3476189

[84] Ellis Wilson, Sudhakar Singh, and Frank Mueller. 2020. Just-in-time Quantum Circuit Transpilation Reduces Noise. *CoRR* abs/2005.12820 (2020). arXiv:2005.12820 https://arxiv.org/abs/2005.12820

[85] Max Wilson, Thomas Vandal, Tad Hogg, and Eleanor Rieffel. 2019. Quantum-assisted associative adversarial network: Applying quantum annealing in deep learning. *arXiv preprint arXiv:1904.10573* (2019).

[86] Xiang Yu, Guoliang Li, Chengliang Chai, and Nan Tang. 2020. Reinforcement Learning with Tree-LSTM for Join Order Selection. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 1297–1308. https://doi.org/10.1109/ICDE48307.2020.00116

[87] Stefanie Zbinden, Andreas Bärtschi, Hristo Djidjev, and Stephan Eidenbenz. 2020. Embedding algorithms for quantum annealers with Chimera and Pegasus connection topologies. In *High Performance Computing*. Springer International Publishing, Cham, 187–206.

[88] Daiwei Zhu, Gregory D. Kahanamoku-Meyer, Laura Lewis, Crystal Noel, Or Katz, Bahaa Harraz, Qingfeng Wang, Andrew Risinger, Lei Feng, Debopriyo Biswas, Laird Egan, Alexandru Gheorghiu, Yunseong Nam, Thomas Vidick, Umesh Vazirani, Norman Y. Yao, Marko Cetina, and Christopher Monroe. 2021. Interactive Protocols for Classically-Verifiable Quantum Advantage. https://doi.org/10.48550/ARXIV.2112.05156