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. We empirically characterise our method on two state-of-the-art approaches (gate-based quantum computing and quantum annealing), and identify speed-ups compared to the best know classical join ordering approaches for input sizes that can be processed with current quantum annealers. However, we also confirm that limits of early-stage technology are quickly reached.

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 achievable solution quality and possible problem size.

In contrast to all prior work on quantum computing for query optimisation and database-related challenges, 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.

SIGMOD '23, June 18–23, 2023, Seattle, WA, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Manuel Schönberger, Stefanie Scherzinger, and Wolfgang Mauerer. 2023. Ready to Leap (by Co-Design)? Join Order Optimisation on Quantum Hardware. In *Proceedings of Proceedings of the 2023 International Conference on Management of Data (SIGMOD '23).* ACM, New York, NY, USA, 16 pages. https://doi.org/XXXXXXX.XXXXXXX

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 IBM [36], Rigetti [66], or D-Wave [50], 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* [58]. It is widely believed, given accepted complexity theoretic assumptions, that quantum systems offer increased computational power over classical systems [4, 7]. Speedups have been proven for multiple quantum algorithms [27, 70], and a seminal experiment [5] 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) [20, 74] is the only application in query optimisation that we are aware of, to the best of our knowledge.

We attribute this to the lack of practical utility of current prototypical QPUs, which are severely limited in various aspects. Given these limits, we cannot expect meaningful results for practical problems on current QPUs, which are classified as so-called *noisy intermediate scale quantum* (NISQ) devices [62]. The goal of this paper is therefore *not* to demonstrate any practically relevant speedups for industrially relevant scenarios (neither was this achieved, to the best of our knowledge, by any of the prior research on using QPUs for databases or any other purpose).

One common assumption in previous work is that the performance of quantum approaches automatically improves with future universal QPUs. Unfortunately, this assumption is not a given [8, 62]: Firstly, universal large-scale, fully error-corrected QPUs are not expected within at least the next decade [34]. Secondly, as we show in this paper, the feasibility of employing QC is closely tied to an alignment of problem and QPU architecture. Even future universal QPUs will not automatically exhibit the required properties to achieve optimal quantum speedups for a specific problem, and will not automatically scale to the required problem dimensions without further ado.

Consequently, we quantitatively argue that one of the most pressing prerequisites to achieving near- and mid-term quantum advantage is the custom co-design of QPUs for specific problems. We outline a concrete path towards near-term DB-QPU utility that goes much beyond mere problem mappings. Rather than waiting on the sidelines, we actively aid QPU development, as we identify problem formulations with careful consideration of QPU properties, and derive design criteria that shape future QPUs as special purpose devices tailored to database applications.

We are the first to follow this QPU co-design principle for database problems. Specifically, 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 [25, 31, 43, 45, 52–54, 56, 57, 72, 75, 81], yet so far not addressed by prior quantum computing research. Since it is not possible to simply deploy existing classical algorithms on QPUs, we reformulate JO into a mathematical description that is unaccustomed in traditional programming, which allows us to use suitable quantum algorithms. Following our co-design principle, we account for specific QPU 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. Our approach is partly based on existing problem transformations, yet it is their non-trivial combination that allows us to identify and address subtle, yet considerable key issues.
(2) We conduct an extensive experimental analysis for join ordering on real QPUs, where we comprehensively consider two state-of-the-art architectures. We analyse the potential of our approach by identifying possible speedups over classical CPU-based state-of-the-art results.
    We also show that problem scalability is limited to small instances on contemporary QPUs. However, our experiments provide directional advice for QPU co-design, and identify specific parameter configurations that provide no issues in a classical context, but have a major impact on the feasibility of using QPUs for join ordering.
(3) We formally derive an upper bound for qubit resource scaling, which allows us to give recommendations for qubit capacitiy sufficient for practical queries on future QPUs.
(4) We specify recommendations for DB-QPU co-design, which has so far not been addressed by prior database research. We conduct numerical simulations for tailored custom QPU designs, and suggest small architectural improvements that can substantially enhance the feasibility of using QPUs for join ordering, and thereby address the bottlenecks identified by our experiments on current QPUs. Our results can be used to accelerate the development of QPUs to reach near-term utility for databases.

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 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 problems. In Sec. 6, we use insights gained from our experimental analysis for QPU co-design, and discuss tailored QPU improvements that greatly enhance near-term DB-QPU utility. We present related work in Sec. 7, and conclude in Sec. 8.

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. For a compact refresher focused on the quantum techniques used in this paper, we provide optional supplementary material. This section summarises the essential knowledge for our quantum approach. Detailed reviews on quantum computing (*e.g.*, Refs. [8, 58]), annealing (*e.g.*, Refs [2, 50]), and the utilised algorithms like QAOA [6, 21, 22, 29, 30, 32] are available in the literature.

Quantum Algorithms. Today, the biggest hurdle for using quantum computing is given by its incompatibility with classical algorithms. To solve JO on QPUs, we have to rely on *quantum algorithms*. Since quantum computation is inherently probabilistic, typical quantum algorithms aim to produce an optimal or near-optimal result with high probability. To obtain a statistical distribution, algorithms are executed *tens to thousands* of times, and an execution is referred to as *shot*.

Quantum algorithms require specific problem encodings. A typical formulation is the *quadratic unconstrained binary optimisation* (QUBO) encoding [9, 46], which (1) only allows *quadratic*, or pairwise, interactions between variables, (2) is *unconstrained* (*i.e.*, does not allow for formulating explicit constraints), (3) only uses *binary* variables, and (4) encodes *optimisation* problems. Physically, we may interpret a QUBO problem as an *energy formula*, where we seek to determine the variable configuration corresponding to the minimum energy that encodes an optimal solution. Mathematically, QUBO problems are 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 $c_{ij} \in \mathbb{R}$ coefficients (it holds that $c_{ij} = c_{ji}$). Finding a JO-QUBO encoding, which we achieve in Sec. 3, allows us to solve JO with quantum algorithms.

The experimental analysis in Sec. 4 considers two algorithms that allow us to solve JO on two fundamentally different QPU architectures: (1) We run the *quantum approximate optimisation algorithm*

(QAOA) [21], which is an iteration-based, hybrid quantum-classical algorithm, to solve JO on *gate-based* QPUs. (2) We solve JO on *quantum annealers* [50], which allows us to draw a comparison of our JO approach to prior results for MQO [74] on this class of machines. While both problems are related to query optimisation, JO and MQO are structurally different, and known MQO to QUBO transformations do not apply for JO.

An important QAOA parameter is the chosen number of gate repetitions, typically denoted p. As shown by Farhi *et al.* [21], the approximation quality improves as p increases. However, even for $p = 1$, QAOA shows promising experimental results for some problems [21, 22], 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 [23], which is seen as clear indicator of possible quantum advantages.

Gate-based QPUs. For DB-QPU co-design in Sec. 6, we are mainly concerned with solving JO with QAOA on gate-based QPUs, as they allow for more design flexibility. As implied by the name, *quantum gates* perform operations on qubits in a *quantum circuit*, and thereby change their state to produce optimal or near-optimal solutions.

While in theory, QAOA solution quality increases for larger p, the required additional gates are detrimental on contemporary QPUs: A fundamental property of a quantum circuit is its *depth* (*i.e.*, the longest sequence of operators in a circuit). Deeper circuits lead to a longer execution time, which is problematic: Unlike classical CPUs, QPU execution time is indirectly limited by the *coherence time* [58] of the hardware, which is, roughly speaking, the time during which quantum properties can be upheld. Increasing circuit depth increases the probability of decoherence errors because of quantum information loss due to environment interaction [65].

Circuit depth is therefore one of the most crucial metrics that we use to evaluate quantum computing feasibility, and is moreover substantially impacted by the properties of the QUBO encoding. For instance, to run the algorithm, the quantum circuit encoding the logical QUBO needs to be *embedded*, or *transpiled*, onto the physically realised QPU architecture, where connections between qubits are scarce. If the encoding requires connections between non-adjacent qubits, missing connections are established by inserting additional gates, thereby increasing depth. This interplay between problem encoding and QPU architecture remains characteristic for quantum computing, and further motivates QPU-DB co-design. We carefully consider such constraints for encoding JO in Sec. 3.

QPU Metrics Summary. Evaluating approaches for QPUs requires mastering a set of new metrics. To evaluate performance on contemporary QPUs, and to analyse the potential achievable by co-designed quantum systems, we consider the following metrics:

(1) The overall *QPU time* required to obtain an optimal or near-optimal result with high probability;
(2) For gate-based QPUs, the *depth of the QAOA circuit* determines execution time, but is also essential w.r.t. other QPU limits;
(3) The *number of qubits* required by the QUBO encoding that needs to be fit onto QPUs.

3 JOIN ORDERING ON QPUS

For the JO-QUBO encoding, it is crucial to find a formulation that is well adapted to the constraints of QPUs. Fig. 1 gives an overview on all steps of our problem reformulation, where we

(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.

In QUBO form, we can embed JO on QPUs, using established heuristic tools, and run quantum algorithms to find solutions. Owing to the probabilistic nature of quantum computing, we nee to run thousands of shots, generating batches of results. In a final postprocessing step, we filter out invalid solutions, and read out the join order from valid ones.

3.1 Formal Model

A JO problem is given by a set of relations $R_0, ..., R_n$, where n_i denotes the cardinality of relation R_i, and a query graph, where each node represents a relation. An edge between relations R_i and R_j is labeled by a predicate p_{ij} with selectivity f_{ij}, where $0 < f_{ij} \leq 1$ [16, 53]. Since our JO-QUBO transformation entails the JO-MILP formulation by Trummer and Koch [75] for classical solvers (but is unrelated to their work on MQO on QPUs), our approach follows their classification for JO: The approach seeks optimal left-deep join trees and allows cross products, without restrictions on the query graph (known to be NP-complete [16]).

We consider the classic cost function C_{out} [16], which is given by $C_{out}(n_i, n_j) := n_i n_j f_{ij}$. Following Cluet and Moerkotte [16] 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 $s_1, ..., s_{i-1}$.

Model Extensions. As we focus on near-term QPU-DB utility, we seek to keep qubit requirements as small as possible, to decrease the load on the limited capacity of contemporary QPUs. Therefore, we restrict our approach to the most basic MILP elements proposed in Ref. [75]. The problem remains NP-complete for this model.

Multiple extensions to the model are discussed in Ref. [75]. However, these require additional variables, which translate to higher qubit requirements. While we refrain from incorporating these for now, they can be freely added in the future alongside QPU advancements, to cover even more complex scenarios. For instance, it is possible to model more sophisticated cost functions for a variety of operators, such as hash join and sort-merge join. Further extensions allow modelling correlations between predicates, which we, for now, assume to be uncorrelated. Finally, an extension to bushy trees may be possible. However, such an extension has yet to be found, and would likely yield a substantial qubit overhead.

3.2 MILP Model and Pruning

In the first step, we encode JO as MILP. The original MILP model in Ref. [75] introduces additional variables to improve comprehensibility, relying on redundant variable elimination by the solver. Since

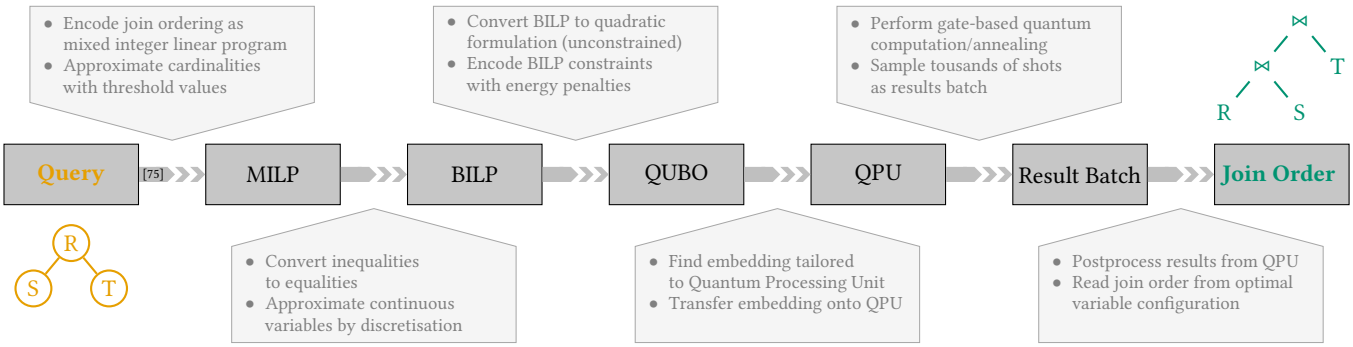

Figure 1: Overview on all steps required to solve join ordering with quantum computing.

additional variables non-linearly translate to additional qubits, we need to reduce the model up-front to retain feasibility on restricted hardware. 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.

3.2.1 Overview. The JO-MILP model contains many variable types and constraints, and may be hard to follow. We nonetheless discuss it, to lay the foundation for subsequent sections. To aid the reader, Table 1 gives an overview of all MILP modelling steps. A running example moreover illustrates each modelling step in detail.

In essence, 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 [17]. For valid solutions, the variables need to satisfy a given set of linear constraints. For JO, MILP constraints are set to enforce a valid configuration of variables that represent various join ordering elements (e.g., join tree leaves, intermediate results and predicates).

Table 1 depicts the four modelling steps: (1) Enforce an unambiguous assignment of join tree leaves, and thereby a valid join order. To evaluate the join order, all subsequent steps contribute towards calculating the join costs. (2) Determine the outer operands, or intermediate results, for all joins. (3) Restrict the use of predicates if relations are missing from join operands. (4) Based on the intermediate results and predicate applicability, approximate the join cardinalities, or costs. The MILP objective function is set to minimise the approximated costs, to find the optimal join order.

Approximation is required since directly encoding the cost function into the MILP objective function is not possible due to the required product operations in C_{out}, which cannot be represented in the linear objective function with linear constraints. To circumvent this issue, Trummer and Koch [75] 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.

3.2.2 Modelling Valid Join Orders. We follow approach and naming conventions of Ref. [75]. 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 if relation t with $0 \leq t \leq T-1, t \in \mathbb{N}_0$ is part of the inner or outer operand of join j with $0 \leq j \leq J-1, j \in \mathbb{N}_0$. We add $2TJ$ such variables to the model (while only tio_{tj} variables for $j = 0$ are relevant for this step, we already add tio_{tj} variables for the remaining joins $1 \leq j \leq J-1$, for the next modelling step). To enforce solution validity, constraints (the latter added for each join j)

$$\sum_t tio_{t0} = 1, \sum_t tii_{tj} = 1, \tag{3}$$

ensure that each leaf of the join tree corresponds to exactly one relation, which enforces a valid join order.

Example 3.1. *Let indexes $0 \leq t \leq 2$ correspond to relations R, S and T respectively. For each join $0 \leq j \leq 1$, we introduce tio_{tj} and tii_{tj}. The constraints unambiguously assign the relations to the join tree leaves: For instance, for the outer operand of join 0, either tio_{R0}, tio_{S0} or tio_{T0} 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. For this running example, we henceforth assume the join order $(R \bowtie S) \bowtie T$, corresponding to the set of active (i.e., with value 1) variables $(tio_{R0}, tii_{S0}, tii_{T1})$.*

3.2.3 Modelling Intermediate Join Operands. For each join $j > 0$ and relation t, the constraint

$$tio_{tj} = tii_{t(j-1)} + tio_{t(j-1)} \tag{4}$$

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 operands of a join. For all except the final join, these constraints are redundantly accounted for by the constraints in Eq. (4). It suffices to include such constraints for the final join and for each relation t:

$$tio_{t(J-1)} + tii_{t(J-1)} \leq 1. \tag{5}$$

Example 3.2. *(cont'd) Consider again our example for the join order $(R \bowtie S) \bowtie T$, represented by the active variables $(tio_{R0}, tii_{S0}, tii_{T1})$. The newly added constraints enforce $tio_{R1} = 1$, since $tio_{R1} = tii_{R0} + tio_{R0} = 0 + 1 = 1$. The same holds for tio_{S1}. As such, the outer operand for join 1 corresponds to the intermediate result after $R \bowtie S$. We add the newly activated variables to our set of active variables: $(tio_{R0}, tii_{S0}, tii_{T1}, tio_{R1}, tio_{S1})$.*

Table 1: Overview of the MILP model, including all variables and constraint descriptions, for each step.

Modelling Step	Var^{s}	Variable Semantics	Required Constraints
① Valid join order	tii_{tj} tio_{t0}	Is *table t in* the *inner operand* of join j / *outer operand* of join 0?	Each join tree leaf represents exactly one relation (Eq. 3).
② Intermediate operands	tio_{tj}	Is *table t in* the *outer operand* of an intermediate join j ($j \geq 1$)?	Include joined relations in all subsequent joins (Eq. 4). A relation cannot be part of both operands of a join (Eq. 5).
③ Predicates	pao_{pj}	Is *predicate p* is *applicable* in the *outer operand* of join j?	A predicate may only be applied if no associated relation is missing (Eq. 6).
④ Cardinality approximation	cto_{rj}	Is *cardinality threshold r* reached by the *outer operand* of join j?	A threshold is added to the join costs if it is exceeded by the logarithmic cardinality of the intermediate operand (Eq. 8).

3.2.4 Modelling Predicates. Consider binary predicates for joining two relations.[1] For each predicate p with $0 \leq p \leq P-1, p \in \mathbb{N}_0$, 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, requiring *both* associated relations as part of the outer operand of join j. For predicate p and join $j > 0$, this is enforced by the constraints

$$pao_{pj} \leq tio_{T_1(p)j}, \qquad pao_{pj} \leq tio_{T_2(p)j}, \tag{6}$$

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

Example 3.3. *(cont'd) So far, (tio_{R0}, tii_{S0}, tii_{T1}, tio_{R1}, tio_{S1}) contains our active variables. We now consider the inclusion of join predicate p_{RS} for relations R and S. One additional variable pao_{01} is required to denote 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 $pao_{01} \leq tio_{R1}$ and $pao_{01} \leq tio_{S1}$ enforce that pao_{01} may only be set to one if both R and S are included in the outer operand of join 1. This is the case in our running example, as $pao_{01} \leq tio_{R1} = tio_{S1} = 1$, allowing $pao_{01} = 1$. The predicate may therefore be applied for 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. Adding the newly activated variable, our set of active variables becomes (tio_{R0}, tii_{S0}, tii_{T1}, tio_{R1}, tio_{S1}, pao_{01}).*

3.2.5 Cost Function and Cardinality Approximation. Finally, based on the prior modelling steps for intermediate operands and predicates, we can approximate the associated costs for the join order. Estimating intermediate cardinalities can be encoded as a MILP problem, based on a logarithmic representation [75]. c_j denotes the logarithmic cardinality for the outer operand of join j by

$$c_j = \sum_t \log(Card(t))\, tio_{tj} + \sum_p \log(Sel(p))\, pao_{pj}, \tag{7}$$

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. For each threshold value r with $0 \leq r \leq R-1, r \in \mathbb{N}_0$ and each join j, a variable cto_{rj} (*Cardinality Threshold reached by Outer operand*) is added to indicate 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 Eq. 2 and only consider intermediate cardinalities, we prune the variables cto_{r0} for join 0. Therefore, $R(J-1)$ variables of type cto_{rj} are required. This is an upper bound, since some cases allow us to prune variables.

To ensure that variables are assigned correct values,

$$c_j - cto_{rj} \cdot \infty_{rj} \leq \log(\theta_r) \tag{8}$$

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. (7).

We observe that in Eq. (8), variable cto_{rj} can be pruned if the maximum value of the logarithmic intermediate cardinality for the 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.

Table 2: Savings of pruned over original MILP model.

	Expression	Saving
Constraint	$tio_{tj} + tii_{tj} \leq 1$	$T(J-1)$
	$pao_{pj} \leq tio_{T_1(p)j}$	P
	$pao_{pj} \leq tio_{T_2(p)j}$	P
	$c_j - cto_{rj} \cdot \infty_{rj} \leq \log(\theta_r)$	$\geq R$
Var	pao_{pj}	P
	cto_{rj}	$\geq R$

Example 3.4. *(cont'd) We conclude our example by approximating the join costs, which, for three relations, merely consist of the cardinality resulting from the first join. We assume input cardinalities as $Card(R) = Card(S) = Card(T) = 100$ and $Sel(p_{RS}) = 0.1$. For this simple scenario, optimal join orders are clearly given by $(R \bowtie S) \bowtie T$ and $(S \bowtie R) \bowtie T$, corresponding to the join costs $Card(R) \cdot Card(S) \cdot Sel(p_{RS}) = 1{,}000$. However, for the MILP approach, we must approximate these costs 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 threshold and for join 1, which has the intermediate result $R \bowtie S$ as the outer operand. As we use the*

[1] We restrict our consideration to uncorrelated predicates for lower qubit requirements, but an extension of the model to correlated predicates is discussed in Ref. [75].

cost function outlined in Eq. (2) and therefore only consider intermediate results, we do not add variables for join 0. For both variables, we add a constraint as given in Eq. (8). For c_j, we obtain $c_j = \log(Card(R))tio_{R1} + \log(Card(S))tio_{S1} + \log(Card(T))tio_{T1} + \log(Sel(p_{RS}))pao_{01} = 2 \cdot 1 + 2 \cdot 1 + 2 \cdot 0 - 1 \cdot 1 = 3$. 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 true intermediate cardinality for $R \bowtie S$. The approximation can be improved by adding more threshold variables. 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 2 summarises savings in variables and constraints by pruning the original MILP model of Ref. [75]. The importance of these savings will be highlighted by the following sections, which describe how variables and constraints translate to qubit requirements and further QPU load, which decisively influences the feasibility on NISQ machines.

3.3 BILP Formulation

To transform MILP to QUBO, as required by QPUs, we build on an intermediate BILP[2] step, since efficient transformations from BILP to QUBO are known [47], 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 additional variables [17], typically called *slack* variables. For instance, converting the constraint in Eq. (8) by adding a slack variable s_{rj} gives

$$c_j - cto_{rj} \cdot \infty_{rj} + s_{rj} = \log(\theta_r). \tag{9}$$

However, a binary domain for s_{rj} is insufficient to ensure the transformed equality is functionally equivalent to the original inequality. Instead, continuous values are required for s_{rj}, which violates the restriction to binary variables. We therefore approximate s_{rj} by discretising with multiple *binary* slack variables, since an integer bounded by C can be expressed using $n = \lfloor \log_2(C) \rfloor + 1$ binary variables [13]. This gives $s_{rj} \approx \omega \sum_{i=1}^{n} 2^{i-1} b_i$, where ω denotes the discretisation precision, and $b_i \in \{0, 1\}$. This results in

$$n = \lfloor \log_2 (C/\omega) \rfloor + 1 \tag{10}$$

binary variables, which, again, leads to a trade-off: More binary variables for the discretisation (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 BILP, QUBO problems cannot include explicit constraints to enforce validity. Instead, we need to ensure that a solution with minimum value inherently corresponds to a valid solution. Lukas [47] provides a conversion that turns BILP problems with N variables into QUBOs with N qubits of the form

$$H = H_A + H_B = \underbrace{A \sum_{j=1}^{m} \left(b_j - \sum_{i=1}^{N} S_{ji} x_i \right)^2}_{H_A} + \underbrace{B \sum_{i=1}^{N} c_i x_i}_{H_B}, \tag{11}$$

where H_A ensures valid, and H_B optimal solutions.

H_A encodes the BILP constraints $Sx = b$. The inner quadratic term evaluates 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 Eq. (11) 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 high penalty weights are known to cause issues like slowdowns [59]. We therefore 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 $A \cdot 1^2 = A$. 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 in H_A to save costs that would otherwise be added in H_B is thereby discouraged, as it will always lead to the same or even larger costs.

Quadratic Contributions. So far, we only addressed how to limit the number of qubits by reducing the number of variables in the problem model. Equally important properties like circuit depth for gate-based QPUs are strongly influenced by the required pairwise, or quadratic, qubit interactions. Consequently, understanding the genesis of quadratic contributions to Eq. (11) is required.

Only H_A, encoding constraints, can entail quadratic interactions. 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. (8)—required for every join and every threshold value—contains the most variables (other constraints contain at most three binary variables, including slack variables). For Eq. (8), 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.

Choosing Approximation Precision. As discussed above, a higher precision via increasing the number of threshold values comes at the expense of higher qubit requirements. It also leads to an increasing amount of quadratic contributions in Eq. (11), as it impacts the

[2]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 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.

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.

QPU limitations should be considered when choosing suitable threshold values. For instance, QPU load can be reduced by setting integer logarithmic thresholds, since these do not require value discretisation, and not only save qubits for encoding larger problems, but also reduce quadratic QUBO contributions. In general, rather than applying an abundance of thresholds, high QPU utility can be best achieved by choosing few, yet carefully selected values, which may be based on available empirical knowledge on intermediate cardinality sizes, to cover a broad range of common scenarios.

The strict necessity to carefully choose approximation precision is unaccustomed from classical approaches, yet characteristic in a quantum context. Our model allows us to freely tune both precisions (thresholds and discretisation precision). This allows us to influence the amount of quadratic contributions, and their impact on the QAOA circuit depth.

3.5 Postprocessing

The final QUBO model in Eq. (11) can be used to solve JO using both, gate-based QAOA, and quantum annealing.[3] From either hardware, we obtain a batch of possible QUBO variable assignments together with the corresponding value of Eq. (11) that indicates solution quality. Each result must be mapped back to the initial JO problem. Since QPUs can deliver invalid solutions owing to hardware imperfections, we need to verify solutions in a postprocessing step.

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

Contemporary QPUs are not expected to solve practically relevant instances of JO. However, despite the limitation to small-sized problems, experiments on actual hardware are beneficial for two reasons: Firstly, they provide the only means to properly evaluate the potential and soundness of our approach, as resource requirements for simulating larger QPUs to tackle practically sized problems scale exponentially, quickly rendering simulations infeasible. Secondly, they provide insights on what limitations should ideally be addressed in future QPUs to gain speedups over classical approaches.

Prior results for query optimisation on QPUs are based on experiments on D-Wave quantum annealers [74]. Therefore, we first analyse JO on D-Wave QPUs, and relate our findings for JO to these prior results for MQO. Like [74], we observe speedups for small problem dimensions, but point out substantial scalability issues. Comparing JO and MQO results provides evidence for the potential of co-designing QPUs for DB problems to ensure scalability.

We then commence to analyse JO on current gate-based QPUs, since these are more suitable for co-design. While these QPUs are similarly limited, and offer no practical utility, we study the impact of specific problem parameters on quantum computing feasibility, and derive insights for co-designing future QPUs for JO.

4.1 Experimental Setup

For quantum annealing experiments, we resort to the D-Wave Advantage [50] system (around 5000 qubits, recently improved with a performance update, Pegasus topology).

For gate-based QAOA, we consider the IBM Q Auckland (27 qubits, Falcon r5.11 topology) QPU. At the time of writing, IBM Q Washington (127 qubits, Eagle r1 topology) is the largest IBM Q system in terms of qubits, but with marked disadvantages in coherence time. Our results for circuit embeddings confirm that these disadvantages outweigh the benefits. We restrict QPU performance analysis to the smaller, more stable Auckland QPU.

Algorithmic Setup. Our approach, implemented in Python, prepares QUBO formulations for a given JO problem, and handles interaction with the QPUs. It relies on library `gurobipy` [28] for formulating MILP and BILP problems. Packages `docplex` [35] (for IBM Q) and `qubovert` [64] (for D-Wave), are used to formulate QUBO representations, which are then processed and passed to the QPU by IBM Qiskit [38] (IBM) and D-Wave Ocean [18].

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 [19]. The QPU runtime, or *annealing time*, is a parameter for D-Wave QPUs. We conduct experiments for varying annealing times t, to study their impact on the solution quality. Since the number of allowed shots decreases with increasing annealing time per shot, we perform 10,000 shots for $t < 10$ μs and 900 shots for $t \geq 10$ μs. Other parameters include the *chain strength*, which groups physical qubits into a chain representing a logical qubit (for detailed information, see Ref. [40]). We experimentally determine suitable chain strengths, depending on JO problem sizes. The exact values can be found in our reproduction package. Remaining parameters are set to their default values.

The Qiskit QAOA 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 $p = 1$, where p denotes the number of QAOA operator repetitions, since larger values for p lead to circuit depths beyond machine capability. Classical optimisation is performed with the Qiskit analytic quantum gradient descent optimiser (AQGD). We sample 1,024 shots for each circuit executed on the QPU. Remaining parameters are set to their defaults.

[3] Note that the *general* structure of a QAOA circuit does not depend on the objective QUBO function [21]. Depicting the resulting quantum circuit for an encoded JO-QUBO is infeasible as it consists of thousands of gates, and would provide no additional insights. We refer interested readers to Ref. [21] and our quantum fundamentals supplement, where the generic QAOA circuit structure is discussed.

Queries. Our goal is to find bounds on the dimension of JO problems for which QPUs determine viable solutions. QPU performance is influenced by a multitude of complex factors, both algorithmic and physical, and deteriorates quickly for increasingly sized problems. That is why we place particular emphasis on the generation of input data: On the one hand, we want to isolate effects of QPU imperfections; on the other hand, we need address scenarios with realistic characteristics.

Input data for the JO problem does not consist of SQL statements, but comprises (statistical) data derived from the relations and predicates involved in an SQL statement. Specifically, input data consists of a query graph (annotated with cardinalities), applicable join predicates with (estimates of) their selectivities, and the threshold values for approximating intermediate cardinalities.

Input data with realistic, representative workload characteristics can be created in two ways: (1) Starting from SQL statements, we can extract required statistical information, and then generate the input query graph. (2) We can directly generate input query graphs alongside a distribution of representative relation cardinalities, predicates and selectivities. The latter method has, for instance, been employed to evaluate the JO-MILP approach of Ref. [75].

Input for the JO problem is, in each case, given by a query graph. To avoid parsing SQL, statistical estimation, and other non-quantum tasks, we apply approach (2). We consider both, multiple generation methods (seminal and recent) for realistic query loads, and state-of-the-art JO algorithms executed on recent classical hardware.

We use (a) the query generation from Ref. [73], which creates queries based on the seminal method of Steinbrunn, Moerkotte and Kemper from VLDB'97 [72] for benchmarking JO approaches. Their implementation supports generating chain, star and cycle queries. Additionally, we (b) generate clique queries (*i.e.*, queries with join predicates for each pair of relations) to compare our approach to recent classical JO results presented by Mancini et al. [48], and generate queries using the exact same method as in their paper.[4]. Since the methods considered in Ref. [48] do not allow for considering cross products, whereas our approach does, we base our comparison on clique graphs. All join pairs in this case are valid, which is equivalent to a cross-join scenario without the need for explicit cross join support in the base method. We find this to enable the fairest comparison.

For cases (a) and (b), we generate sets of 20 queries for different problem size scenarios, where we consider between two and five relations (we find below that larger queries exceed the limited capacity of our subject QPUs).

Additionally, we (c) generate input data with cardinalities, selectivities and threshold values hand-crafted that try to minimise the expected influence of imperfections of current QPUs on solution quality: As pointed out in Sec. 3, current QPUs require discretising continuous variables, at the expense of reducing QPU capacity for increasing precision. We restrict this set of input data to integer logarithmic cardinalities, predicate selectivities and threshold values, as they do not require value discretisation. Data and code for all methods are supplied in the reproduction package.

[4]We appreciate that the authors have provided their source code to allow for an exact reproduction.

Limitations of larger IBM Q machines, as explained above, necessitate restricting 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 in Sec. 6. Compared to IBM Q, the D-Wave Advantage system allows for embedding larger queries, although limits are also reached quickly, as shown below.

Even with the restriction to three relations, we can study JO problems with varying properties on IBM Q systems. 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, to identify parameter-specific bottlenecks.

4.2 Experimental Results

We next discuss the experimental evaluation on state-of-the-art QPUs, focusing on (a) run-time, (b) result quality, and (c) maximal problem sizes (qubit count, circuit depth).

4.2.1 *Join Ordering on D-Wave.*

Experimental Results. For QA, runtime is a parameter: If it is too small, the probability for a non-optimal or invalid solution increases; if it is too large, this is obviously counter to speedups. The overall QPU runtime essentially depends on the *annealing time per shot* Δt and the *number of shots* n required until an optimal solution is found; Tab. 3 shows observed experimental values. Since n is a statistically distributed quantity, we compute the average value $\bar{n}$ to determine the average runtime $T = \bar{n} \cdot \Delta t$. As expected intuitively, increasing Δt reduces $\bar{n}$. T is minimal for a (short) annealing time of $\Delta t = 0.5\mu s$, which we use in the following experiments.

To evaluate the potential of our JO-QPU approach, we need to compare it against an established baseline. However, the criteria for fair QPU-CPU comparisons are subject of scientific debate [51]. Therefore, we can neither demonstrate universal speedups over all parameter ranges, nor show practical utility. At the same time, such comparisons are vital to demonstrate the potential of QPU advantages to the DB community.

Mancini *et al.* [48] have, at SIGMOD'22, presented a massively parallel, efficient JO algorithm that they compare to some of the fastest known state-of-the-art JO approaches (Postgres, DPCCP [54], DPE [31], DPSub [52], DPSize [52]—it is beyond the scope of this paper to discuss these classical algorithms in detail). To the best of our knowledge, this is the only reference in the literature with a sole focus on join-ordering that is (a) very recent, (b) addresses

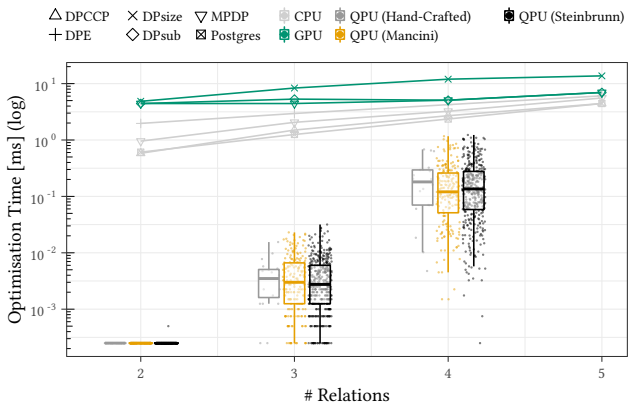

Figure 2: Potential analysis of QA-JO by comparing QPU optimisation time results with best-in-class CPU/GPU approaches obtained at SIGMOD'22 [48].

an input data scenario comparable to ours, (c) covers a broad selection of JO algorithms, and (d) gives quantitative optimisation time measurements. We compare our QPU results inferred for all input data scenarios to their corresponding classical results.

Fig. 2 compares our quantum results against their published classical numbers. Since our approach is inherently stochastic, we provide the runtime distribution as compared to the mean values reported by Mancini *et al.* [48] for state-of-the-art classical algorithms. The middle vertical line in the boxplots represent our observed median results.

To identify the impact of discretisation costs on QPUs, we compare solution time for queries with integer logarithmic values, which do not require any variable discretisation, against solution time for Steinbrunn and Mancini queries, which capture realistic scenarios, where we set the discretisation precision to two decimal positions, which we determined as the best choice (*i.e.*, resulting in the highest solution quality) for the considered queries after preliminary testing.

For small problem dimensions, we observe pronounced *speedups*—encompassing multiple orders of magnitude!—over *all* classical algorithms provided in Ref. [48]. While not considering these results direct evidence, as we stress again, for the practical feasibility of our approach, we do see them as a promising indication.

These speedups are almost equally pronounced for both, integer logarithmic queries, and the more general Steinbrunn and Mancini queries, and the impact of discretisation costs seems negligible for the considered small-scale problems: QPUs can attain optimal solutions despite the higher QPU load due to discretisation. Neither do the specific query cardinalities and selectivities seem to exert much impact on QPU performance.

It therefore suffices to consider one set of input data in the following, and we choose integer logarithmic queries. This allows us to identify upper bounds independent of QPU discretisation costs, and also reduces the pecuniary impact of QPU access somewhat.

While we find the results for small-scale queries very promising, great care needs to be applied when interpreting the observations, and especially when extrapolating to larger instances. Table 3 lists the probabilities of reaching valid and optimal solutions for the more common chain, star and cycle graphs with up to five relations (minimal discretisation precision, one inequality threshold value). Solution quality is not much influenced by query graph topology, but declines quickly with an increasing number of relations (the impact of annealing times is minimal, which agrees with previous observations in the literature [44, 69]). In general, it suffices if one annealing shot delivers a (near-)optimal solution; this is the case for three and four relations. However, for five relations, almost *none* of the 900 shots include a valid, and none an optimal solution—independent of query graph type and annealing time. This is likely attributable to the magnitude of perturbations in a a fully utilised annealer [51].

Another impediment is that technical issues like communication between QPU and CPU must be taken into account. While these could be reduced to negligible levels, they can—depending on the actual setup—exceed the sampling time by orders of magnitude larger than the sampling time. This must be taken into account when evaluating indiscriminately optimistic claims on possible QPU speedups, as seen in the literature.

We conclude that while our results indicate substantial potential, solving JO problems with more than four relations is infeasible for *current* D-Wave QPUs. Unfortunately, simulating large, noisy quantum systems as would be required to provide an estimate of the performance of future QAs in the absence of real hardware is itself an NP-complete problem that is beyond reach [8] (if simulating such systems were feasible, there would be no need to build quantum annealers in the first place).

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

Query Graph	Δt [μs]	3 Relations Valid	3 Relations Opt.	4 Relations Valid	4 Relations Opt.	5 Relations Valid	5 Relations Opt.
Chain	10	31.00%	8.69%	1.23%	0.07%	0.12%	0%
	100	37.01%	8.22%	1.53%	0.19%	0.07%	0%
	1000	41.25%	10.30%	1.78%	0.14%	0.15%	0%
Star	10	-	-	2.16%	0.33%	0.03%	0%
	100	-	-	2.01%	0.29%	0.02%	0%
	1000	-	-	2.48%	0.43%	0.04%	0%
Cycle	10	27.52%	11.46%	3.32%	0.41%	0.02%	0%
	100	31.10%	14.36%	3.77%	0.46%	0.02%	0%
	1000	34.58%	16.89%	4.21%	0.36%	0.04%	0%

Context. Let us set these results into context with the findings of the only previous study on using a quantum annealer for query optimisation. Trummer and Koch [74] also compare a quantum approach against classical benchmarks, but for a different problem (MQO instead of JO). They find advantages for larger problems which albeit far from reaching practical dimensions, surpass solvable problem sizes in our case.

A fundamental difference between thee approaches explains this discrepancy: In contrast to using heuristic tools to determine

embeddings for JO, the MQO-QUBO is mapped to the physical hardware using a hand-crafted embedding. Most importantly, the fixed patterns of MQO agree well with the QPU connectivity structure.

It is not possible to enjoy these benefits for JO (and most other problems): It is only suitable for problems with predictable and fixed structures independent of the input, whereas pairwise qubit interactions in JO depend on the problem instance. Restricting problems to fixed structures also restricts generality and scalability. Also, the embedding problem is NP-complete [49, 82], and any manual approach is deemed to quickly become infeasible. However, the MQO results suggest a pronounced link between solution quality and problem alignment with quantum hardware.

Instead of trying to align different problems with *unique* requirements to QPUs with *fixed* architecture, we instead suggest to *tailor* QPUs to DB problems via DB-QPU co-design. Since gate-based QPUs offer more design flexibility [8] they are better suited to co-design than quantum annealers. Consequently, we next analyse the performance of our approach on gate-based IBM-Q QPUs, and then consider how co-design can improve their performance toward practical utility in Sec. 6.

4.2.2 Join Ordering on IBM Q. For IBM Q hardware, we analyse circuit depths for varying JO problems and QPUs in addition the the observables considered in the previous section. The depth of a circuit on a gate-based QPU crucially impacts feasibility; with increasing depth, quantum coherence is lost, and result quality degrades.

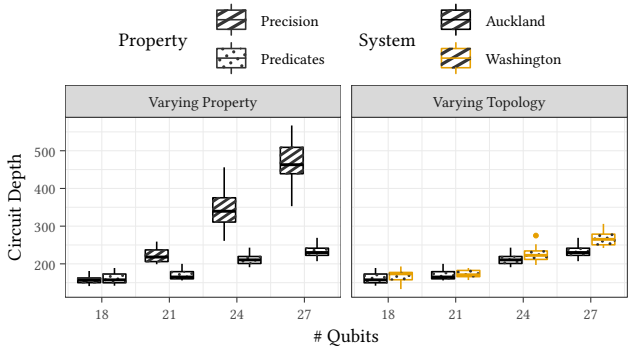

Figure 3: Circuit depths of various scenarios, IBM Q devices.

Circuit Depths. While the number of required qubits is a metric for problem *size*, the circuit depth can be seen to quantify the impact of specific problem *parameters*. These require varying interactions between qubits, and lead to different depths. Therefore, two problems with identical size (in qubits) may produce substantially different circuit depths, which can likewise substantially impact the solution feasibility on a QPU.

This scenario is depicted in Fig. 3, which 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. The left hand side of Fig. 3 compares circuit depths for problems with three relations and one threshold value, but different numbers of predicates and approximation precisions. By varying the discretisation 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. 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 discretisation 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 results suggest that qubit connectivity offered by current QPUs is a major impediment to scalability, since a higher discretisation precision demands more qubit interactions. Our co-design simulations of improved QPU architectures in Sec. 6 confirm this hypothesis.

The right hand side of Fig. 3 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,[5] 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). Given these parameters, the approximate maximum depth d of a circuit to not exceed $T1$ or $T2$ is given by $d = \lfloor \min(\mathrm{T1}, \mathrm{T2})/g_{\mathrm{avg}} \rfloor$. We thereby get $d_A = 293$ for the Auckland QPU and $d_W = 168$ for the Washington QPU. Even for scenarios with less discretisation precision, the maximum depth is quickly reached for Washington, where we therefore cannot expect meaningful results, outweighing its size benefits. This, again, shows that waiting for QPUs with higher qubit capacity does not guarantee more favourable system-global properties, and rather motivates us to better align problems and topologies.[6] Even for Auckland, problems with a high precision exceed this depth. For experimentally analysing QPU performance, we therefore only analyse JO problems at the minimum precision, using the Auckland QPU.

QAOA Performance. Following the stability considerations

Table 4: Solution quality for n QAOA iterations (three relations).

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

[5] IBM QPUs undergo periodic recalibration operations to optimise T_1 and T_2, so the values are not stable across different experiments.
[6] *Quantum volume* [55] has been suggested as more balanced measure that weighs various characteristics of QPUs; interestingly, it is 64 for both QPUs in our experiments.

from above, we restrict our performance experiments to the 27-qubit Auckland QPU, which allows for working JO problems with three relations before the available qubit capacity is exhausted. Table 4 shows QAOA results for 1,024 shots, for queries with increasing predicate numbers, and provides the fraction of measurement shots that represent valid or optimal solutions.

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 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 for one QAOA iteration. For 0 predicates, $t_s = 77.9$ms. The time increases to $t_s = 113.70$ms for three predicates. Given that the average number of classical QAOA iterations steps (using the COBYLA optimiser) required until convergence is 28.5, it is obvious that this type of QPU is currently unable to compete with the classical approaches summarised in Fig. 2, and is 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 [42]) be addressed by error-corrected, perfect QPUs.

To progress towards earlier QPU utility, we pursue two strands: First, we derive an upper bound on the amount of required qubits for JO; this allows us to make suggestions for qubit capacities sufficient for realistic queries, yet also leaves room to address additional limitations like circuit depth. Using insights gained from our experimental analysis, we then address the question of which properties of QPUs should be improved to progress towards practical utility via co-design (recall that we have argued above why gate-based QPUs are a better candidate for co-design efforts than annealers).

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. (9), an upper bound for s_{rj} is given by

$$\log(\theta_r) + \infty_{rj} \geq \log(\theta_r) + cto_{rj}\infty_{rj} - c_j = s_{rj}. \tag{12}$$

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. (8), 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. (12), 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 $\forall k < l : Card(k) \geq Card(l)$.*

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 discretisation 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 Eqs. (5),(6). 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. (8). Following from Lemma 5.1 and Eq. (10), 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 $n = n_{\text{pec}} + n_{\text{sl}}$, or qubits, as

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

□

6 DB-QPU CO-DESIGN FOR JOIN ORDERING

We now commence to deriving recommendations for future QPU designs customised towards serving as co-processors in databases, using insights gained from our experimental and formal analysis.

6.1 Qubit Recommendations

Given the formal analysis in Sec. 5, we first address problem scalability by deriving recommendations for qubit capacity. Fig. 4 thereby visualises upper qubit bounds, based on our formal analysis, for a variety of JO problems with up to 64 relations and different discretisation precisions. We choose cycle queries, as they require one additional join predicate compared to chain and star graphs.

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.

While qubit capacity for solving the largest problems considered with classical MILP solvers [75], where queries with 60 relations are joined, is difficult to achieve in the near-term (requiring a QPU with more than 20,000 qubits), such queries are on the upper end of the spectrum w.r.t. JO problem sizes. Aiming for near-term DB-QPU utility, we instead recommend aiming for qubit capacity sufficient for typical query loads, where we consider queries as contained in the JO benchmark by Leis et al. [45] as representatives. These join between 3 and 16 relations. Our method requires approx. 1,000 qubits (depending on approximation and discretisation precision) for queries with up to 13 relations, which is roughly equal to the typical query load. Vendor roadmaps predict such QPUs within very few years [34]. This provides an optimistic outlook w.r.t. the impact of qubit requirements on the problem scalability.

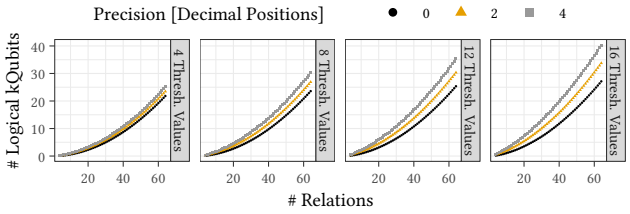

Figure 4: Upper bounds for logical qubits for JO with varying approximation (thresh. values) and discretisation precisions.

6.2 Topology Recommendations

Based on our experimental results, which emphasized the need to address QPU limitations beyond just qubits, we now analyse the feasibility of solving JO problems on hypothetical improved QPU topologies, depending on the resulting circuit depths.

Scenarios. We extrapolate new topologies in three ways: By increasing the qubit capacity (and thereby, solvable problem dimensions) based on a structural extension of the available connectivity graph, by adding additional qubit connections, and by using different quantum gate sets. 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.* [72], using the query generator code by Trummer [73]. We consider problems with two threshold values, and minimal discretisation precision (*i.e.*, $\omega = 1$).

Size Extrapolation. We consider baseline designs from IBM [36] (127 qubit Washington), IonQ [37], and Rigetti [66] (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. This physical principle features stable, universally connected qubits, but slow gate times. Superconducting qubits offer less connectivity and stability, but faster gates [8, 58]. The repeating patterns of IBM and Rigetti topologies allow for straightforward extrapolation to larger numbers of qubits.

Density Extrapolation. Our experimental results suggest a large impact of connectivity limitations on JO feasibility, indicating great potential for improvement. As such, 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 in topological proximity are more likely 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, but close qubits. Given the set of connections C_δ between qubits with distance δ, we uniformly add connections sampled from C_δ starting with $\delta = 2$, until reaching the desired density, or all elements of C_δ have been added. Restart from $C_{\delta+1}$ in this case.

Results. Fig. 5 shows the depths of the quantum circuits for combinations of randomly generated join ordering problems, 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 network with density 1 is obviously impossible using planar graphs.

Therefore, rather than improving qubit numbers beyond capacities sufficient for the bulk of practical queries, we heavily recommend improving qubit connectivity, since even a small amount of extra connections, which we deem achievable by future QPUs, can substantially impact the utility of QPUs for JO problems. Similar observations, albeit not as pronounced, hold for Rigetti.

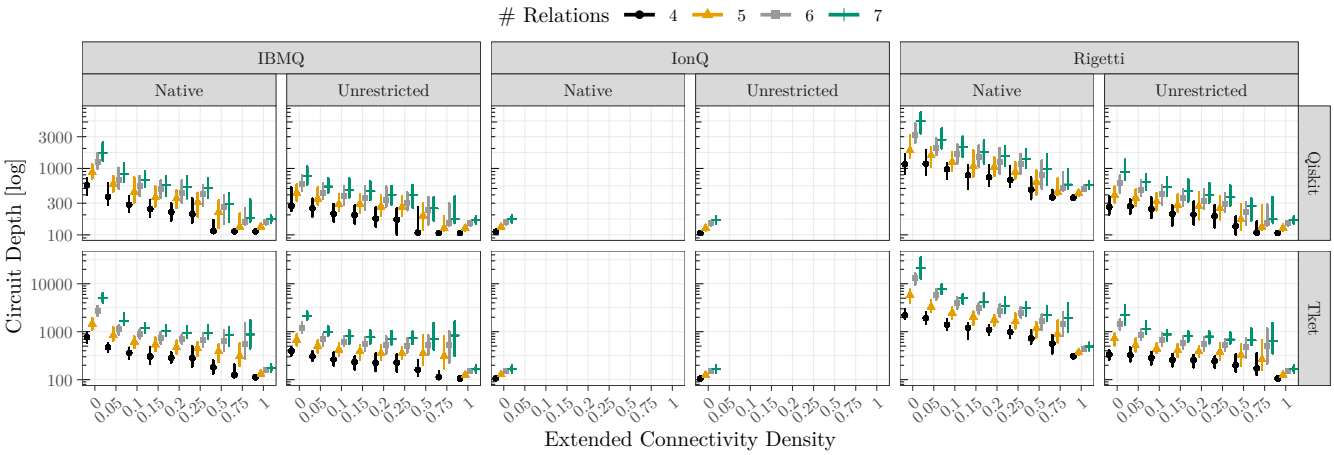

Figure 5: Circuit Depths for hypothetical future QPUs on varying join order instances. Experiments are performed with different query graph topologies, but these are not visualised since no relevant differences arise.

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 has gained recent attention for QPUs (see, *e.g.*, Refs. [3, 12, 77]), for instance, for noise reduction [79] or approximate circuits [78]. Our experiments consider two (sufficiently mature) transpilation approaches, Qiskit [38] and tket [71]. 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. Both produce comparable results for complete meshes (IonQ), so we conclude heir capabilities are similar for boundary cases, but need to be carefully evaluated otherwise.

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 [24], 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.

7 RELATED WORK

Join ordering is among the most studied problems in database research [25, 31, 43, 45, 52–54, 56, 57, 72, 75, 81]. While join ordering on modern hardware, e.g., GPUs [48], is intensively being researched, no published work exists for JO with QCs, to the best of our knowledge. Very few endeavours address the use of QCs in databases, in stark contrast to the growing interest in quantum computing in other fields [14, 15, 33, 39, 41, 60, 61, 63, 67, 68, 76, 80].

DB transaction scheduling was studied by Groppe and Groppe [26], and Bittner and Groppe [10, 11]. The MQO problem was analysed by Trummer and Koch [74] 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, where they achieved promising results by aligning specific problems to the available hardware. Fankhauser *et al.* [20] addresses MQO for gate-based QPUs.

Our method differs in several ways. JO and MQO problems are structurally different. While JO is concerned with deriving optimal join orders for a query plan, MQO seeks optimal plan combinations for multiple queries. We cannot reuse existing transformations for MQO, and propose a novel method to obtain a JO-QUBO formulation. The JO-QUBO transformation entails formulating JO as MILP, following Ref. [75], where Trummer and Koch solve JO with classical MILP solvers. However, we substantially adjust their formulation to arrive at a JO-QUBO encoding, and are the first to solve JO on QPUs, even gaining some quantum advantage.

Ref. [20, 74] analysed MQO on existing QPUs, without practical utility. For near-term utility, we derive DB-QPU co-design recommendations, and identify improvements tailored to JO.

8 DISCUSSION AND CONCLUSION

Quantum computing promises—grounded on theoretical insights and guarantees [8], but also based on first experimental results [5]—speedups for computational problems over classical approaches. The technology is apt for many aspects of database systems, including query optimisation. So far, the use of QCs for DB is extremely underexplored. Yet, our work is Janus-faced: On the one hand, we show that *current* NISQ-systems are far away from producing practical benefits, or handling realistically sized instances. This 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, given the commercial importance of databases.

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. Co-design efforts should be based on step-wise empirical refinement leveraging expert knowledge from QC, DB, and systems engineering.

We lay the ground: Our approach enables the experimental exploration on two major classes of QCs, and identifies key factors that inhibit scalability and practical utility. We provide directions on designing QPUs favourable for JO. Nonetheless, many open research problems remain, from efficient circuit generation respecting noise, to more targeted extensions of topologies that transcend our semi-stochastic approach, to considering alternative information encoding schemes that might alleviate discretisation problems.

Yet such approaches are challenged by major physical and algorithmic obstacles, and require truly interdisciplinary research.

Contributions. MS (≈ ¾) and WM (≈ ¼) wrote the paper. SS suggested JO as problem of interest. WM and MS designed, and MS implemented and performed the experiments. WM and MS performed data analysis and visualisation.

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] 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

[3] 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

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

[5] 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

[6] 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

[7] 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

[8] 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

[9] 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.

[10] 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

[11] 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.

[12] 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

[13] 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.

[14] 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

[15] 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

[16] 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.

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

[18] 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/

[19] 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

[20] 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

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

[22] 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

[23] 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

[24] 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

[25] 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

[26] 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

[27] 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.

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

[29] 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

[30] 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

[31] Wook-Shin Han and Jinsoo Lee. 2009. Dependency-Aware Reordering for Parallelizing Query Optimization in Multi-Core CPUs. 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, 45–58. https://doi.org/10.1145/1559845.1559853

[32] 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

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

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

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

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

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

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

[39] 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

[40] D-Wave Systems Inc. 2020. Programming the D-Wave QPU: Setting the Chain Strength. https://www.dwavesys.com/media/vsufwv1d/14-1041a-a_setting_the_chain_strength.pdf

[41] 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.

[42] 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

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

[44] 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

[45] 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.

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

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

[48] Riccardo Mancini, Srinivas Karthik, Bikash Chandra, Vasilis Mageirakos, and Anastasia Ailamaki. 2022. Efficient Massively Parallel Join Optimization for Large Queries. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 122–135. https://doi.org/10.1145/3514221.3517871

[49] 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.

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

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

[52] Andreas Meister and Gunter Saake. 2020. *GPU-accelerated dynamic programming for join-order optimization.* Technical Report. https://www.inf.ovgu.de/inf_media/downloads/forschung/technical_reports_und_preprints/2020/TechnicalReport+02_2020-p-8268.pdf

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

[54] Guido Moerkotte and Thomas Neumann. 2006. Analysis of Two Existing and One New Dynamic Programming Algorithm for the Generation of Optimal Bushy Join Trees without Cross Products. In *Proceedings of the 32nd International Conference on Very Large Data Bases* (Seoul, Korea) *(VLDB '06)*. VLDB Endowment, 930–941.

[55] 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

[56] 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

[57] 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

[58] 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.

[59] 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.

[60] 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

[61] 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

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

[63] 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

[64] 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

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

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

[67] 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

[68] 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

[69] 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

[70] 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.

[71] 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

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

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

[74] 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.

[75] 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.

[76] 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

[77] 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

[78] 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

[79] 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

[80] 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).

[81] 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

[82] 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.