Quantum-Inspired Digital Annealing for Join Ordering

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

Immanuel Trummer
Cornell University
Ithaca, NY, USA
itrummer@cornell.edu

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

ABSTRACT

Finding the optimal join order (JO) is one of the most important problems in query optimisation, and has been extensively considered in research and practise. As it involves huge search spaces, approximation approaches and heuristics are commonly used, which explore a reduced solution space at the cost of solution quality. To explore even large JO search spaces, we may consider special-purpose software, such as mixed-integer linear programming (MILP) solvers, which have successfully solved JO problems. However, even mature solvers cannot overcome the limitations of conventional hardware prompted by the end of Moore's law.

We consider *quantum-inspired* digital annealing hardware, which takes inspiration from quantum processing units (QPUs). Unlike QPUs, which likely remain limited in size and reliability in the near and mid-term future, the digital annealer (DA) can solve large instances of mathematically encoded optimisation problems *today*. We derive a novel, native encoding for the JO problem tailored to this class of machines that substantially improves over known MILP and quantum-based encodings, and reduces encoding size over the state-of-the-art. By augmenting the computation with a novel readout method, we derive valid join orders for each solution obtained by the (probabilistically operating) DA. Most importantly and despite an extremely large solution space, our approach scales to practically relevant dimensions of around 50 relations and improves result quality over conventionally employed approaches, adding a novel alternative to solving the long-standing JO problem.

PVLDB Reference Format:
Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. Quantum-Inspired Digital Annealing for Join Ordering. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/lfd/vldb24.

1 INTRODUCTION

Query optimisation involves solving complex NP-hard problems, where obtaining sufficiently good solutions can be difficult. One of the most fundamental query optimisation problems is given by the join ordering (JO) problem, whose general form features an extremely large search space. While exhaustive dynamic programming (DP) approaches can obtain optimal solutions for small-scale queries, they fail once queries reach moderate or large sizes. Hence, query optimisers tend to substitute polynomial-time heuristics for DP approaches once queries reach certain dimensions. However, these typically involve trade-offs to reduce the search space. Accordingly, solution quality can suffer in comparison to optimal solutions for less restricted JO search spaces. Large queries regularly occur in real-world workloads [9, 38, 55]. For such complex queries, the difference in execution time between good and average plans can reach several orders of magnitude [38]. In particular when executing such queries on large data, investing time into sophisticated query optimisation approaches is well worthwhile if it avoids sub-optimal plan choices due to heuristic optimization.

Special-Purpose Solvers. To efficiently explore even large JO solution spaces, we can, for instance, rely on highly optimised special-purpose solvers: Trummer and Koch [53] formulated JO as a mixed integer linear programming (MILP) problem, allowing the use of mature MILP solvers, which have been optimised over decades, and are likely to further improve in the future. However, the performance of such optimisation software is naturally tied to the executing hardware (HW). As the development of conventional general-purpose HW begins to stagnate with the end of Moore's law, even the most highly optimised software will likely fail to mitigate the bottlenecks arising due to physical limitations.

Special-Purpose Hardware. As a logical next step, we consider the potential of *special-purpose* HW, whose design is tailored to specific algorithms. To address the limits of conventional systems, research in many fields, including database (DB) research [12, 31, 35, 59], has begun to explore the potential of HW-SW co-design, which may be among the only adequate means to accomplish significant performance increases going forward. Tailored systems provide a sustainable, energy-efficient alternative to general-purpose systems, for which energy efficiency is an increasing challenge.

Quantum Hardware. One of the most highly anticipated kinds of special-purpose HW are quantum processing units (QPU), which can inherently address the limitations faced by the manufacture of conventional systems, as they are able to exploit quantum-mechanical phenomena to achieve speedups impossible for conventional systems. To solve optimisation problems on QPUs, they have to be transformed into specific *mathematical encodings*. Schönberger *et al.* [41] derived such an encoding for the JO problem, allowing the use of QPUs for JO optimisation. However, contemporary QPUs, given their early-stage prototypical nature, are restricted in several

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

ways, and hence unable to harvest the potential of quantum computation for practical problems. As such, only small-scale queries can be optimised on contemporary quantum systems. In addition, the performance of QPUs is heavily tied to the conformance of the mathematical problem encoding and the QPU architecture [41], which further motivates HW-SW co-design.

To arrive at QPU-DB maturity, we next outline two methods by which we improve over Schönberger *et al.* [41]. Specifically, we (a) consider *quantum-inspired HW*, rather than pure QPUs, to determine QPU potential *today*, and to assess whether co-design efforts are worthwhile. Further, we (b) derive a novel JO encoding that is *highly tailored* to quantum and quantum-inspired HW.

Quantum-Inspired Hardware. Instead of waiting for mature QPUs, we rely on quantum-inspired HW, such as the high-performance Fujitsu Digital Annealer (DA) [3, 13], which mimics the principles of QPUs to solve mathematically encoded optimisation problems. The DA's performance can be considered a lower bound for actual quantum HW, and provides first insights into the potential of mature QPUs. However, we do not consider quantum-inspired systems as temporary substitutes, as they allow us to solve large NP-hard optimisation problems beyond the capabilities of conventional HW, and can be brought into immediate use. Indeed, speedups over conventional methods by orders of magnitude have been reported in other domains [3, 28], motivating DA evaluation for DB issues.

Tailored Encoding. We derive a novel, native JO encoding, tailored to quantum and quantum-inspired HW that improves over the *baseline* JO encoding proposed by Schönberger *et al.* [41], which is a faithful transformation of the JO-MILP formulation by Trummer and Koch [53], and hence inherits all MILP limitations. Firstly, our novel encoding more efficiently approximates solution cost, by refining the approximation method proposed in [41] with previously unconsidered quadratic operations. Secondly, applying entirely novel strategies, our encoding (1) more efficiently encodes solution validity, (2) ensures validity for *all solutions*, unlike the baseline, (3) drastically reduces variables, and (4) conforms significantly better to DA HW, outperforming the baseline encoding [41] in all aspects. Thereby, we exhaust the full capabilities of the Fujitsu DA, to solve large JO problems far beyond conventional sizes.

Contributions. In detail, our contributions are as follows:

(1) We derive a novel, native encoding for JO problems tailored to special-purpose quantum and quantum-inspired hardware. We thereby address several drawbacks of the existing MILP and QPU encodings, optimising the scalability of our method.
(2) We propose a novel readout method tailored to our JO encoding, allowing us to derive valid join orders for each solution obtained by the annealer. In contrast to both, the MILP method and baseline encoding used on QPUs, where optimisation may fail to yield any solution, our method thereby guarantees to obtain a specifiable number of join orders.
(3) We analyse the resource requirements for our novel encoding, and demonstrate a significant reduction in variables over the baseline QUBO encoding, thereby further improving scalability.
(4) We compare our method against a variety of competing approaches. Using the quantum-inspired Fujitsu Digital Annealer, we experimentally demonstrate the practical utility and scalability of our novel annealing method on large queries joining up to 50 relations, which far exceeds typical query dimensions. Our approach identifies complex solutions within a large JO search space, which often beat the best join orders obtainable within the restricted search space of competing methods.

Table 1 further highlights our contributions, by summarising key criteria where our novel, native method for the Fujitsu DA improves on the existing MILP approach by Trummer and Koch [53] and the baseline QPU method by Schönberger *et al.* [41].

Table 1: Comparison of the MILP [53], baseline JO QPU [41], and our novel JO digital annealing (DA) approach.

	MILP	QPU	DA
Special-purpose HW support	✗	✓	✓
Mature SW/HW	✓	✗	✓
Efficient native encoding	✓	✗	✓
Predictable optimisation time	✗	✓	✓
Guaranteed result	✗	✗	✓

The remainder of this paper is structured as follows: We explain the properties of the Fujitsu DA hardware, and fundamentals on the required encoding, in Sec. 2. We describe our JO model in Sec. 3. We derive a novel JO-encoding for quantum and quantum-inspired HW in Sec. 4. We discuss our readout method in Sec. 5, and experimentally analyse our DA method in Sec. 6. Finally, we discuss related work in Sec. 7 and conclude in Sec. 8.

2 DIGITAL ANNEALING FUNDAMENTALS

Solving combinatorial optimisation problems is typically complex, given highly non-linear solution landscapes. To avoid local minima, simulated annealing takes inspiration from physical annealing of crystalline solid: Its search strategy is dynamically altered based on a gradually cooling temperature. In this section, we outline, and contrast, the three annealing paradigms *simulated annealing* (Sec. 2.1), *quantum annealing* (Sec. 2.2), and quantum-inspired *digital annealing* (Sec. 2.3). Finally, contemporary quantum and quantum-inspired HW require a specific quadratic problem encoding, which we describe in Sec. 2.4.

2.1 Simulated Annealing

The original *simulated annealing* (SA) algorithm is a metaheuristic for solving combinatorial optimisation problems [3, 22]. Algorithm 1, taken from Aramon *et al.* [3], shows the overall steps of the SA algorithm. Starting from a random initial state at a high temperature T, T is gradually lowered by performing monte carlo (MC) updates, which involves flipping bits of variables. Ideally, after the algorithm terminates, the resulting low-temperature variable configuration corresponds to an optimal, or near-optimal, solution. Rather than only running the algorithm once, a large batch of runs, or *shots* (typically involving hundreds or thousands of runs), is performed. The number of runs can be specified, and provides a trade-off between solution quality and computation time.

Algorithm 1 Simulated Annealing

```
for each run do
    initialise to random initial state
    for each temperature do
        for each MC sweep at this temperature do
            for each variable do
                propose a flip
                if accepted, update the state and effective fields
        update the temperature
```

2.2 Quantum Annealing

While SA seeks to determine the variable configuration minimising an optimisation problem, quantum annealing (QA) attempts to determine the *ground state* of a *Hamiltonian* operator, which describes the energy of a quantum system [1], and encodes the optimisation problem to be solved. While both approaches address similar problems, the processes behind are quite different.

2.2.1 Annealing Process. Beginning with an initial Hamiltonian whose ground state is known, the Hamiltonian is gradually transitioned towards the actual problem Hamiltonian representing the optimisation problem. A sufficiently slow interpolation is known preserve the ground state, in accordance to the adiabatic theorem [1]. Once the process terminates, one can obtain an optimal solution by reading out the ground state. Quantum effects such as *tunneling* enable QA to avoid local minima in concave energy landscapes, providing a potential computational advantage over classical approaches.

2.2.2 State-of-the-art. The first QA systems have become commercially available; the most advanced ones like the *D-Wave Advantage* feature over 5,000 physical quantum bits (qubits) [29]. Current systems only support limited connectivity between qubits—to solve problems with many interacting variable pairs, sets of physical qubits are combined into logical qubits via *chains of qubits*.

This leads to multiple issues: (1) Combining physical qubits into chains effectively reduces the amount of available qubits representing variables; (2) problems are required to be *embedded* onto the hardware graph of the quantum annealer, which is an NP-complete problem itself [27, 58]; and (3) longer chains increase the likelihood of *chain break errors* (mismatched qubits in a chain).

Solution quality further degrades via the influence of *noise* and *quantum decoherence* (*i.e.*, a gradual decay of the quantum state due to interactions with the environment). The culminated impact of these issues severely limits the practical utility of contemporary quantum annealers, as previously analysed by Schönberger *et al.* [41] for the JO problem. Given these issues, practical quantum systems likely remain out of reach for the foreseeable future.

2.3 Digital Annealing

The Fujitsu digital annealer (DA) method is inspired by quantum phenomena as discussed above, yet is implemented using classical technology. Its capabilities can be seen as *lower bound* on future QPUs, but also as HW accelerator for computational tasks.

2.3.1 State-of-the-art. In contrast to available QA systems, the Fujitsu DA features HW with fully connected bits. Several DA generations are available: Second generation systems provide full connectivity for 8,192 bits [13], supporting problems with up to 8,192 variables. In contrast, third and fourth generation devices feature a *software intervention layer* for automating problem conversion steps and penalty weight calculation (details discussed below) [36]. However, due to additional optimisation time overhead, and to make our approach compatible with future quantum or classical annealing devices, we consider the second generation DA for our paper, which is closest in nature to current quantum devices.

As a classical device, the DA is unaffected from detrimental effects like quantum decoherence. While it cannot utilise quantum effects for speedups, its massive parallelisation capacity, combined with a SA variant tailored to exploit these, allows the DA to solve complex optimisation problems that may exceed conventional HW.

2.3.2 Annealing Process. The Fujitsu DA operates based on the SA algorithm, modified to include quantum-inspired enhancements to exhaust its full parallelisation capabilities. Its specific steps are featured in Algorithm 2, taken from Aramon *et al.* [3].

Algorithm 2 Digital Annealing

```
initial_state ← an arbitrary state
for each run do
    initialise to initial_state
    E_offset ← 0
    for each MC step (iteration) do
        if due for temperature update, update the temperature
        for each variable j, in parallel do
            propose a flip using ΔE_j − E_offset
            if accepted, record
        if at least one flip accepted then
            choose one flip uniformly at random amongst them
            update the state and effective fields, in parallel
            E_offset ← 0
        else
            E_offset ← E_offset + offset_increase_rate
```

While SA starts off a randomised state, each DA execution commences in a known state , thus avoiding to initialise effective fields for each randomised run. Using a *parallel trial scheme* and the highly connected DA HW, flips for all individual variables are evaluated in parallel in each MC ste. If at least one flip is recorded as accepted, one out of all recorded flips is uniformly selected and applied, updating the effective fields of all neighbors in parallel in *constant time regardless of the number of neighbors*. Plain SA considers only single bit flips at a time, and the time required for updating neighboring fields grows linearly in the number of neighbors.

Instead of tunneling to escape local minima, the DA employs a *dynamic offset* escape method to increase the chance of accepting bit flips if none were accepted in the previous iteration. For more details on DA, we refer to Aramon *et al.* [3].

2.4 Problem Encoding

It is not possible to run arbitrary code on quantum and quantum-enhanced systems. Instead, problems require specific encodings, and have to be cast as *quadratic unconstrained binary optimisation* (QUBO) problems, which (1) only allow *quadratic interactions* between bits, or variables, (2) are *unconstrained*, and hence do not support including *explicit* constraints, (3) only support *binary* variables, and (4) solve *optimisation* problems. Physically, QUBOs may be interpreted as an energy formula, where the minimum energy solution corresponds to an optimal solution of a problem reduced to QUBO. They are given by 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 $c_{ij} = c_{ji} \in \mathbb{R}$ coefficients.

Since enforcing valid solutions is not possible using explicit constraints, our goal is to transform the JO problem in such a way that minimising its energy formula inherently ensures valid solutions, by penalising any invalid solution with a positive energy cost. Achieving this goal allows us to optimise queries on the Fujitsu DA. We next discuss the specific characteristics of our considered JO problem class, for which we present a novel transformation method into a JO-QUBO in Sec. 4.

3 JOIN ORDERING MODEL

A variety of classifications exist for the JO problem. As we are interested in evaluating the aptitude of DA for the NP-complete case, it is important to pick sufficiently hard instances.

A JO problem is classified by (a) *problem input*, (b) *solution space*, and (c) a *cost function* assigning costs to each solution. We classify our approach according to these elements below.

3.1 Problem Input

The JO input is given by a *query graph* $Q = (V, E)$, where each node $v_i, 0 \leq i \leq |V|$, corresponds to a relation r_i with cardinality n_i to be joined, and an edge e_{ij} represents a join predicate p_{ij} for relations i and j. Edges are further labeled by a respective join selectivity f_{ij}, where $0 < f_{ij} \leq 1$. While some JO methods require graphs with certain properties, *e.g.*, acyclic query graphs [18, 23], our DA approach allows *general query graphs* with no restrictions.

Query Graph Archetypes. Since the query graph shape impacts JO behavior and complexity, we consider commonly used archetypes, as exemplified in Fig. 1.

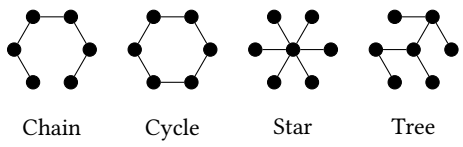

Figure 1: Examples for common query graph archetypes.

These archetypes represent scenarios commonly found in JO literature and real workloads (*i.e.*, star queries correspond to OLAP applications). Further, the shape impacts JO behavior: For star queries, the central relation is ideally joined within the first join operations, to allow the application of join predicates. Optimal join orders for star graphs hence only rarely include *cross products* (defined below), which may be useful for other scenarios, *e.g.*, linear chain and cycle graphs, to skip and postpone joining unfavourable relations.

We consider these archetypes for our experimental evaluation in Sec. 6, to generate large, synthetic queries for common scenarios, suited to comprehensively assess the scalability of our method.

3.2 Solution Space

A solution to the JO problem is given by a *join tree*, where leaf nodes correspond to base relations, and intermediate nodes represent join operations. Therefore, the size of the solution space depends on (a) the possible join tree *shapes*, and (b) the *possible leaf permutations*.

Join Tree Shapes. Tree shapes can be broadly summarised by two scenarios: Trees with *no restrictions* are referred to as *bushy* trees, contrasting *linear* trees, where at least one base relation is an operand for each join. The latter includes left-deep, right-deep and zig-zag trees. The restriction to linear trees is a common heuristic, motivated by a more efficient exploration of a reduced solution space. Like the existing MILP [53] and QA approaches, our DA method considers left-deep join trees.

Leaf Permutations. A join tree is moreover characterised by its assignment of leaf nodes, with the number of possible assignments given by $n!$ for a query joining n relations. Given the large amount of possibilities, many JO approaches consider a restricted set of permutations, excluding the possibility of *cross products*, *i.e.*, joining two intermediate join trees not sharing any join predicates. However, as we will show empirically, the cost overhead for excluding cross products can be quite substantial. Our DA method hence considers cross products, enabling cheaper plans in many cases.

3.3 Cost Function

Finally, a *cost function* assigns costs to each join tree. For our DA method, we consider the classic cost function $C_{out}(n_i, n_j) = f_{ij} n_i n_j$ [8]. Applied on a sequence of relations, it evaluates join orders based on the sizes of intermediate results. Following Cluet and Moerkotte [8], the cost function for a sequence s of relations $s_1, ..., s_n$ is given by

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

where $|s_1 ... s_{i-1}|$ denotes the result size after joining $s_1, ..., s_{i-1}$.

Typically, sets of join predicates feature *correlations*, which beget discrepancies between the cost provided by C_{out}, and the true cost including such correlations. While we plan to include them in future research, this paper focuses on the base model without correlations.

Summary. Our DA method (a) considers *general* query graphs, (b) *left-deep* join trees, (c) *cross products*, and (d) the cost function C_{out}. This JO classification features an extremely large solution space with factorial growth in the number of relations, and is moreover known to be NP-complete [8], making the task of obtaining sufficiently good solutions for large queries challenging.

Limitations. We consider the NP-complete JO classification outlined above sufficiently complex for providing a first evaluation

of the aptitude of special-purpose HW such as the DA. However, the restriction to left-deep join trees and the inability to consider predicate correlations are clear limitations. While extensions to general, bushy trees are desirable, their encoding remains challenging, given the constraints of contemporary hardware. The limitation to quadratic interactions requires a considerable overhead, and greatly impacts the scalability of the method for current systems [42].

While co-designing annealing HW tailored to such other classes of JO problems may be promising, we first need to understand whether such approaches are worth pursuing. This requires insights into the aptitude of DA for JO problems conforming well to devices available *today*, which is the goal of this paper.

4 QUBO ENCODING

To solve JO on the Fujitsu DA, we have to transform the JO base problem into QUBO form. For the existing JO-QUBO, Schönberger *et al.* [41] apply a faithful transformation of the MILP method [53] into QUBO. However, the resulting JO-QUBO inherits all limitations of the original MILP method, without making use of extended QUBO functionality not supported by MILP solvers. Rather than transforming an existing encoding, we propose an encoding tailored to the QUBO formalism, which fully exhausts hardware capabilities.

To encode JO as QUBO, we have to specify variables and constraints in such a way that (1) we receive *valid join trees* as solutions, and (2) the annealer seeks out solutions *evaluated favorably by our cost function*. Hence, we show a new, lightweight method to enforce valid join trees in Sec. 4.1, and propose a novel method for encoding solution cost in Sec. 4.2, inspired by the baseline encoding [41], relying, however, on quadratic operations unsupported by MILP, and unconsidered by the baseline. Finally, we combine the intermediate QUBO encodings into an overarching JO-QUBO in Sec. 4.3.

4.1 Encoding Valid Join Trees

We begin by introducing variables expressing the join tree. Let the binary variable roj_{rj} (*Relation is Operand for Join*), introduced for each relation r and join j, indicate whether r is an operand for j. Our goal is to ensure that value assignments for these variables conform to valid join trees. Hence, let us next consider the two validity conditions that must hold for variable assignments:

(a) Starting at two relations, the number of relations serving as operands for a join strictly increases by one with each additional join. The validity of this condition is shown by Theorem 4.1.

(b) Once used as an operand for a join j, a relation must moreover serve as an operand for all joins directly or indirectly succeeding j. This condition follows from the very definition of a join.

Theorem 4.1. *The number of relations serving as operands for a join strictly increases by one with each additional join.*

Proof. The validity of the theorem can be quickly deduced from the definition of a left-deep join tree, which considers at least one input relation as an operand for each join. The first join takes two input relations as operands, whereas any intermediate join considers all relations used for its preceding join, and one not previously joined input relation as operands. Starting at two relations, the number of relations used as operands hence increases by one for each additional join. □

Next, we show how to encode both conditions in QUBO. Recall that the annealer seeks out variable configurations minimising the energy of the QUBO formula. We can enforce valid solutions by designing QUBO terms that increase the energy for invalid solutions, and only evaluate to 0 for valid solutions.

4.1.1 Condition (a). We begin with condition (a), which can be enforced by the following QUBO term:

$$HV_a = \sum_{j=1}^{J} \left(b_j - \sum_{r=1}^{R} roj_{rj} \right)^2 ,$$

where $b_j = j + 1$. To minimise the energy, the inner sum has to correspond to the value b_j for each join j, such that the quadratic term evaluates to 0. This requires the correct amount of *roj* variables to be active, depending on the join index j. For instance, for $j = 1$ (*i.e.*, the very first join), $b_j = j+1 = 1+1 = 2$. As such, two variables $roj_{r_1 j}$ and $roj_{r_2 j}$, corresponding to any pair of input relations (r_1, r_2), need to be active to avoid an energy penalty, whereas all remaining variables must equal 0. Therefore, as the annealer minimises the energy, exactly two relations are selected as operands for join 1. The same applies mutatis mutandis for all subsequent joins, for increasing numbers of required relations.

Example 4.1. *To illustrate the effectiveness of HV_a, let us consider a query joining three relations A, B and C, using two joins i and j, where i precedes j. Accordingly, we introduce six variables roj_{Ai}, roj_{Bi}, roj_{Ci}, roj_{Aj}, roj_{Bj}, roj_{Cj}, to express the join tree. Let us further consider a solution with active variables $roj_{Ai} = roj_{Bi} = roj_{Ci} = 1$. Clearly, this solution is invalid, since all three relations have been assigned as operands to join i, and no relation is assigned to j. Accordingly, the energy penalty $HV_a = (2 - (roj_{Ai} + roj_{Bi} + roj_{Ci}))^2 + (3 - (roj_{Aj} + roj_{Bj} + roj_{Cj}))^2 = (-1)^2 + (3)^2 = 10$ is added.*

In contrast, successful energy minimisation via annealing may activate the variables $roj_{Ai} = roj_{Bi} = roj_{Aj} = roj_{Bj} = roj_{Cj} = 1$, where $HV_a = (0)^2 + (0)^2 = 0$. Indeed, this solution corresponds to the valid join order $(A \bowtie B) \bowtie C$.

4.1.2 Condition (b). While HV_a enforces the correct number of relations to be present for each join, it alone does not yet ensure valid variable assignments, since a relation may be part of a join j, yet not be present for all joins succeeding j. As such, we proceed by encoding condition (b) as follows:

$$HV_b = \sum_{r=1}^{R} \sum_{j=2}^{J} roj_{rj-1}(1 - roj_{rj}).$$

HV_b enforces that $roj_{rj-1} = 1$ always implies $roj_{rj} = 1$, as otherwise, the inner term in H_b evaluates to 1, contributing a positive value to the overall energy. Therefore, if relation r is an operand for join $j - 1$, it must also serve as an operand for the succeeding join j.

Example 4.2. *(cont'd) We continue our example of joining three relations A, B and C, using joins i and j, where i precedes j. Let us again consider the invalid solution with active variables $roj_{Ai} = roj_{Bi} = roj_{Ci} = 1$. Disregarding the violation of condition (a), which was outlined above, this configuration moreover violates condition (b), since neither of the three relations is present for join j, after selected as an operand for the preceding join i. Accordingly, energy*

penalty $HV_b = roj_{Ai}(1 - roj_{Aj}) + roj_{Bi}(1 - roj_{Bj}) + roj_{Ci}(1 - roj_{Cj}) = 1(1-0) + 1(1-0) + 1(1-0) = 3$ *is added.*

In contrast, for the valid variable assignment $roj_{Ai} = roj_{Bi} = roj_{Aj} = roj_{Bj} = roj_{Cj} = 1$*, representing the join order* $(A \bowtie B) \bowtie C$*, no penalty is added, as* $HV_b = 1(1-1) + 1(1-1) + 0(1-1) = 0$.

Finally, we combine the discussed penalty terms into an overarching validity term for left-deep join trees

$$HV = HV_a + HV_b.$$

We later combine this validity term with the QUBO term for cost assignment, which we discuss next.

4.2 Encoding Join Order Costs

Having enforced join tree validity, our next goal is to assign a join tree a corresponding cost value, in accordance to our cost function C_{out}, which evaluates intermediate join result sizes. Herein lies one of the greatest hurdles for optimising join orders with MILP or annealing, as neither approach supports the product operations required by C_{out}. Instead, MILP is confined to linear operations, whereas QUBO supports both, linear and quadratic operations.

For their MILP approach, Trummer and Koch [53] therefore substitute sums of logarithmic cardinalities and selectivities for the required product operations, as the logarithm of a product equals the sum of logarithms of its factors. Based on the logarithmic intermediate results, they approximate actual cardinalities using an arbitrary number of threshold values, as illustrated below.

For the existing JO-QUBO encoding, Schönberger *et al.* [41] conducted a faithful transformation of the original MILP approach, and hence incorporated this approximation method. However, as we discuss in detail below, this approximation does not conform well to contemporary annealing devices. To improve on this *baseline cost approximation* approach, we hence propose a *novel quadratic cost approximation* method, which makes use of quadratic operations unsupported by MILP, and not utilised by the existing JO-QUBO, and is hence natively tailored to contemporary annealers.

We first discuss how to encode the logarithmic cost of intermediate join results. Next, we describe the baseline cost approximation as applied by Schönberger *et al.* [41] in detail. Finally, we discuss our novel quadratic cost approximation method.

4.2.1 Encoding Logarithmic Cost. To encode logarithmic costs for a join, we require information on both, relations and join predicates present for the join. The former are already encoded by variables *roj*. Similarly, let the binary variables paj_{pj} (*Predicate is Applicable for Join*) denote whether predicate p can be applied for join j. For valid configurations of *paj* variables, we need to enforce that $paj_{pj} = 1$ only holds if both relations associated with predicate p are operands for join j. Deriving this information from the corresponding *roj* variables, we achieve this by adding the following term:

$$H_p = paj_{pj}(2 - roj_{Rel_1(p)j} - roj_{Rel_2(p)j}),$$

where $Rel_i(p), 1 \leq i \leq 2$, corresponds to the first or second relation associated with p. As such, setting $paj_{pj} = 1$ without activating both $roj_{Rel_1(p)j}$ and $roj_{Rel_2(p)j}$ leads to an energy penalty.

Based on the *roj* and *paj* variables, the following function gives the logarithmic intermediate cardinality for join j:

$$LogIntCard(j) = \sum_{r=1}^{R} LogCard(r)\, roj_{rj} + \sum_{p=1}^{P} LogPredSel(p)\, paj_{pj},$$

where $LogCard(r)$ and $LogPredSel(p)$ are variable coefficients, providing the logarithmic cardinality for relation r and the logarithmic selectivity for predicate p respectively.

Example 4.3. *(cont'd) To illustrate the logarithmic cost calculation, we continue our example of joining three relations A, B and C with joins i and j, where i precedes j. Let* $n_A = n_B = n_C = 2$ *denote the logarithmic cardinalities for the three relations. Let us further assume two join predicates,* p_1 *for relations A and B, and* p_2 *for relations B and C, with logarithmic selectivities* $f_1 = f_2 = -1$*. Accordingly, we add four variables* paj_{1i}*,* paj_{1j}*,* paj_{2i} *and* paj_{2j}*.*

Recall the valid variable configuration $roj_{Ai} = roj_{Bi} = roj_{Aj} = roj_{Bj} = roj_{Cj} = 1$*, expressing the join order* $(A \bowtie B) \bowtie C$*. To minimise cost, the annealer seeks to activate as many paj variables as possible, to apply their corresponding predicates. Ideally,* $paj_{1i} = paj_{1j} = paj_{2i} = paj_{2j} = 1$*. However, since* $paj_{2i}(2 - roj_{Bi} - roj_{Ci}) = 1(2-1-0) = 1$*, the energy penalty* $H_p = 1$ *is added, as relation C is not an operand for join i, invalidating the activation of* paj_{2i}*. Hence, to minimise cost, the annealer will only activate the remaining variables* $paj_{1i} = paj_{1j} = paj_{2j} = 1$[1].

Finally, the logarithmic cardinality resulting from join i is then given as $LogIntCard(i) = n_A roj_{Ai} + n_B roj_{Bi} + n_C roj_{Ci} + f_1 paj_{1i} + f_2 paj_{2i} = 2 + 2 + 0 - 1 - 0 = 3$*, and for join j as* $LogIntCard(j) = n_A roj_{Aj} + n_B roj_{Bj} + n_C roj_{Cj} + f_1 paj_{1j} + f_2 paj_{2j} = 2 + 2 + 2 - 1 - 1 = 4$.

Having derived the logarithmic intermediate result sizes for the join tree, our next goal is to approximate their actual cardinalities.

4.2.2 Baseline Cost Approximation. To apply the baseline cost approximation following Trummer and Koch [53], a set of threshold values is added to the model. The more thresholds are applied, the more accurate the cost approximation becomes. Let the binary variables trj_{tj} (*Threshold is Reached by Join*), introduced for every threshold value $1 \leq t \leq T$ and join $1 \leq j \leq J$, denote whether a logarithmic threshold value $\log(\theta_t)$ has been exceeded by the logarithmic size of the intermediate result produced by join j. If exceeded, the threshold value θ_t will be added to the overall costs accordingly. This is achieved by the following cost term:

$$HC = \sum_{t=1}^{T} \sum_{j=1}^{J} trj_{tj}\theta_t. \tag{3}$$

In MILP, the following inequality constraint ensures that $trj_{tj} = 1$ if $\log(\theta_t)$ has been reached by join j:

$$LogIntCard(j) - trj_{tj} \cdot \infty_{tj} \leq \log(\theta_t). \tag{4}$$

If $LogIntCard(j) > \log(\theta_t)$, the only way to satisfy the inequality is to activate trj_{tj}, which subtracts the sufficiently large constant ∞_{tj}.

[1] For simplification, we omitted one detail in our example, as the benefit of activating the invalid predicate variable may, in general, still outweigh the induced energy penalty, and doing so may hence yield the minimum energy. The solution is to properly balance the validity and cost terms, as we discuss in detail in Sec. 4.3.

However, expressing the inequality given by Eqn. 4 in QUBO poses an issue, since inequality operations are not inherently supported. Therefore, Schönberger *et al.* [41] convert the inequalities to equalities, by adding a continuous variable s_{tj}, which gives

$$LogIntCard(j) - trj_{tj} \cdot \infty_{tj} + s_{tj} = \log(\theta_t).$$

This engenders further issues, as QUBO only supports binary, rather than continuous variables. Following Schönberger *et al.* [41], we hence need to approximate s_{tj} as $s_{tj} \approx \omega \sum_{i=1}^{n} 2^{i-1} b_i$, discretising the continuous variable using multiple binary variables b_i. Further, $\omega = (0.1)^d$ denotes a discretisation precision, where d is the number of allowed decimal positions. We require $n = \lfloor \log_2(c_{j_{max}}/\omega) \rfloor + 1$ binary variables for the discretisation of s_{tj}, where $c_{j_{max}}$ is the maximum logarithmic cardinality possible for join j. Hence, the smaller ω (*i.e.*, the higher the discretisation precision), the larger the number of required variables. More detailed explanations on the discretisation are provided in Schönberger *et al.* [41].

Example 4.4. *(cont'd) To illustrate the baseline cost approximation, we continue our example of a query joining three relations A, B and C with joins i and j, where i precedes j. We further consider the previous variable configuration representing the join order $(A \bowtie B) \bowtie C$, and the logarithmic costs $LogIntCard(i) = 3$ and $LogIntCard(j) = 4$ for joins i and j, as calculated in Example 4.3. Let $\theta_1 = 100$ and $\theta_2 = 1000$ denote the threshold values used to approximate the actual cost. Accordingly, we introduce four variables trj_{1i}, trj_{2i}, trj_{1j} and trj_{2j}. Attempting to avoid cost, the annealer seeks to leave trj variables inactive. Ideally, $trj_{1i} = trj_{2i} = trj_{1j} = trj_{2j} = 0$. However, since $LogIntCard(i) = 3 > 2 = \log(\theta_1)$, the threshold is exceeded, and trj_{1i} must be active to avoid a penalty. The same holds for trj_{1j}, as $LogIntCard(j) = 4 > 2 = \log(\theta_1)$, and trj_{2j}, since $LogIntCard(j) = 4 > 3 = \log(\theta_2)$. However, no penalty is induced for $trj_{2i} = 0$, as $LogIntCard(i) = 3 \leq 3 = \log(\theta_2)$. To minimise cost, the annealer will hence leave trj_{2i} inactive, bringing the total cost to $HC = \theta_1 trj_{1i} + \theta_1 trj_{1j} + \theta_2 trj_{2i} + \theta_2 trj_{2j} = 100 + 100 + 0 + 1000 = 1200$. Since the approximated cost is quite far from the real cost, given by $10^3 + 10^4 = 1000 + 10000 = 11000$, we observe that the accuracy of the approximation highly depends on the applied thresholds.*

Example 4.4 demonstrates the great impact of the threshold values on the approximation accuracy. While it is possible to increase the approximation quality by adding more thresholds, larger models typically demand longer optimisation times. Moreover, when using special-purpose HW, rather than conventional systems with large memory, we have to consider its size limitations, which restrict the number of allowed optimisation variables. To accomplish scalability to large queries, we hence seek to optimally use the available variables, and next discuss a novel, more space-efficient approximation technique. It is tailored to contemporary annealers by using quadratic operations that are unsupported by MILP.

4.2.3 Novel Quadratic Cost Approximation. To increase the efficiency of the cost approximation, we can rely on quadratic operations not utilised by the existing JO-QUBO by Schönberger *et al.* [41]. Specifically, we can substitute the following quadratic cost function for the baseline cost method and inequality operation required to verify if a threshold has been exceeded:

$$HC = \sum_{t=1}^{T} \sum_{j=1}^{J} \theta_t \left(Buffer_{tj} - LogIntCard(j) \right)^2, \tag{5}$$

where the term $Buffer_{tj}$ is given by

$$Buffer_{tj} = \sum_{i=1}^{N} (2^{i-1} s_{tij}), \tag{6}$$

where s_{tij} is a binary variable. We set N in such a way that $Buffer_j$ can assume any value up to $\log(\theta_t)$.[2]

For any $LogIntCard(j) \leq \log(\theta_t)$, there exists a variable configuration minimising the formula by setting $Buffer_{tj} = LogIntCard(j)$, causing the inner quadratic term in HC to vanish. However, if $LogIntCard(j) > \log(\theta_t)$, $Buffer_{tj}$ cannot balance out $LogIntCard(j)$, thus increasing the cost. For integer values, the smallest possible excess is $Buffer_{tj} - LogIntCard(j) = -1$ (*i.e.*, the threshold is exceeded by one order of magnitude). Accordingly, $\theta_t \cdot (-1)^2$ is added to the cost. Increasing threshold excess increases the added cost value.

Example 4.5. *(cont'd) We complete our running example by illustrating the quadratic cost approximation on the two join orders $JO_1 = (A \bowtie B) \bowtie C$ and $JO_2 = (A \bowtie C) \bowtie B$, for our query joining relations A, B and C with joins i and j, where i precedes j. For join i, the former produces logarithmic costs 3, whereas the latter produces logarithmic cost $n_a + n_c = 2 + 2 = 4$, as no predicate is applicable. Since costs for the final join j are invariant w.r.t. the join order, we restrict our example to the costs produced by join i, where we consider the single threshold value $\theta_1 = 100$, with $\log(\theta_1) = 2$, for the cost approximation[3]. Since either join order exceeds the threshold, applying the baseline approximation does not discriminate between them, as the threshold is added to the cost in either case.*

In contrast, let us now consider the effect of our quadratic cost approximation. We add a sufficient number of variables to the term $Buffer_{1i}$, such that it can assume any value up to $\log(\theta_1) = 2$. Hence, it can prevent any cost penalty if the threshold is not exceeded. However, since either join order cost exceeds the threshold, the buffer cannot balance out the logarithmic cost of join i. To minimise cost, the annealer sets $Buffer_{1i} = 2$, and we receive $\theta_1(Buffer_{1i} - LogIntCard(i))^2 = 100(2-3)^2 = 100$ for JO_1, and $100(2-4)^2 = 400$ for JO_2. Hence, even though the threshold is exceeded by both join orders, minimising the quadratic approximation formula produces the less costly JO_1.

While existing methods for MILP [53] and the baseline annealing method [41] only test *if* a threshold has been reached, our novel approach takes the *magnitude* of the excess into account, as demonstrated by Example 4.5. A single threshold value serves as a much more sophisticated discriminator between join orders differing in cost by orders of magnitude, which is common for JO. This allows us to increase the approximation quality for a set of thresholds, or

[2] We may alternatively apply a one-hot encoding for $Buffer_{tj}$, which allows arbitrary threshold values at the cost of needing one variable for each possible value $Buffer_j$ may assume. In contrast, our applied binary encoding can express *specific* thresholds much more efficiently, but only allows thresholds as values $(2^i \cdot \omega) - 1$, where the term $\omega = (0.1)^d$ denotes the discretisation precision with d decimal positions.

[3] To simplify our example, we apply no restrictions on the choice of thresholds, whereas for our own implementation, we apply the variable-efficient binary encoding scheme as in Eqn. 6, and hence restrict thresholds to values $(2^i \cdot \omega) - 1$ for any $i \geq 1$, where ω is the discretisation precision.

reduce the number of thresholds required for sufficiently accurate results, which is particularly beneficial for special-purpose HW with strict size limitations.

Further, while our novel cost approximation refines the baseline proposed by Trummer and Koch [53], its fundamental implications for optimal solutions remain unchanged. Hence, we may, for hypothetical perfect devices (disregarding, *e.g.*, size limits of existing HW), introduce an arbitrary amount of thresholds, to make our approximation arbitrarily precise, hence allowing configurations that guarantee optimal solutions when minimising the QUBO formula.

Having derived both, QUBO terms for ensuring valid join orders, and terms for approximating their costs, the final step is to combine them into an overarching JO-QUBO.

4.3 The Complete Encoding

Based on the validity terms discussed in Sec. 4.1, and the cost terms derived in Sec. 4.2, we can construct the complete JO-QUBO H:

$$H = A(HV + H_p) + HC.$$

The term A is a penalty weight that amplifies the penalty for violating validity constraints, such that any potential cost saving in HC can never beget amortisation of a penalty induced by violating H_{val}, or by activating invalid predicates, violating H_p. However, A cannot be set arbitrarily large, which would beget issues such as slowdowns [39]. We hence seek the minimum weight for A that engenders sufficiently large penalties. This weight is given by the largest possible value C_{max} that may be assumed by the term HC, which a solver may seek to avoid by violating a constraint in HV.

For our novel quadratic approximation, following Eqn. 5, our cost function HC is maximised if every threshold for every join is exceeded by the largest possible intermediate result size c_j. In that case, $Buffer_{tj} = \log(\theta_t)$, and $C_{max} = \sum_{t=1}^{T} \sum_{j=1}^{J} Thres(t) \left(\log(\theta_t) - c_j\right)^2$. We hence set $A = C_{max} + \epsilon$, where ϵ is some small value, to properly balance our validity and cost terms.

While these aprioristic bounds ensure that constraint violations never lead to a more desirable overall energy outcome, it is unlikely that a single constraint violation would beget cost savings akin to these worst-case bounds. To avoid the negative impact of high values for A, it is therefore beneficial to empirically determine suitable values for A, rather than applying these lower bounds.

Resource Scaling. Finally, for archetypical cycle queries, Fig. 2 compares our novel encoding against the baseline encoding [41] w.r.t. the required amount of *mandatory* variables n_{var} (without thresholds for approximation). Our novel encoding requires $n_{novel} = (R + P)(J - 1)$ for queries joining R relations with J joins, using P join predicates, which provides a lower qubit bound[4].

The baseline reaches capacity limits of the second generation DA for queries joining 40 relations, requiring roughly 8,000 variables, and maximum query size further decreases due to thresholds. In contrast, using our novel encoding, optimising queries with up to 50 relations is easily possible on today's systems while still leaving capacity for thresholds, demonstrating the variable efficiency of our novel method. Further comparisons, including scenarios with thresholds, can be found in Appendix A.

[4]Following Schönberger *et al.* [41], their encoding requires $n_{baseline} = 2RJ + R + (3P)(J - 1)$ such mandatory variables.

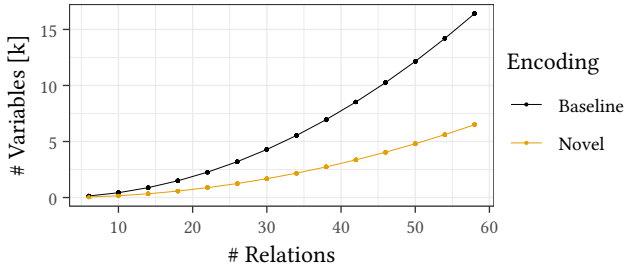

Figure 2: Mandatory variable scaling for cycle queries, comparing the baseline JO-QUBO [41] and our novel encoding.

5 READOUT

Our JO-QUBO formulation *encodes* queries to allow the use of quantum and quantum-inspired annealing. In each case, devices deliver a *solution bitstring* that contains value assignments for all variables. A flawless annealing run would return only valid solutions where assignments adhere to validity terms of Sec. 4. This would make reading out the join order straightforward.

However, annealing solutions often contain errors, whose frequency depends on the properties of the device. These errors manifest in the form of *bitflips*, which add ambiguity to an annealing solution and hence complicate the readout process. A concrete example for a solution containing bitflips is discussed below. The manner in which such bitflips are addressed substantially impacts the practical utility of the overall annealing method.

Baseline Readout Method. The baseline QA approach [41] only considers a solution as valid if the assignment of certain variables leaves no ambiguity, discarding all other solutions returned by the annealing device. However, quantum annealers are prone to errors of various kinds, which can engender a large amount of bitflips. For queries joining four relations, their method considers merely one to three percent of all solutions as valid. This discards a large amount of *possible* starting points for solutions.

However, even though a solution may contain ambiguity for certain variables, it may still include valuable information for constructing a good join order. Rather than entirely discarding such solutions, we seek to extract as much information as possible from a single solution, to construct a valid join order in any scenario, and doing so for as many solutions as possible.

Novel Readout Method. Hence, for our novel QUBO encoding, we propose a readout strategy that encompasses maximum efficiency, deriving valid join orders for any solution returned by the annealing device. This provides our method with an advantage not only w.r.t. the original approach by Schönberger *et al.* [41], but moreover regarding any JO method that can potentially time out without returning any solution at all, including many dynamic programming approaches as well as the MILP method by Trummer and Koch [53].

For reading out a solution, we only need to consider variables that assign relations to their respective joins, and can disregard all other variables. Hence, we are only concerned with variables roj_{rj}, for each relation $1 \leq r \leq R$ and join $1 \leq j \leq J$. We further consider a bitstring b_r that contains the *roj* variable assignments for a relation r. For instance, given a query with three joins, the

bitstring 111 expresses that relation r is part of all joins, whereas the bitstring 001 expresses that r is only part of join 3. Either of these bitstrings corresponds to valid configurations of *roj* variables.

Consider the bitstring 1101 for four joins. The bitstring is erroneous, given that relation r is selected for join 1 and 2, yet missing for join 3, and once again present for join 4. This configuration violates one constraint, as our encoding penalises solutions where roj_{r2} does not imply roj_{r3}, yet may still occur owing to stochastic nature of annealing. Intuitively, we can identify the third bit as the likely candidate for a bitflip, as the relation has already been selected for both, the first and second join. Correcting the bitstring to 1111 is more plausible than to 0001.

In general, we assume that the more early joins a relation r is assigned to in accordance to an annealing bitstring b_r, the greater the plausibility that r is selected by the annealer to be joined earlier rather than later, and hence should appear early in the resulting join order. To capture this, we first assign weight w_j to each join j, where $w_j = J - i$, $0 \leq i \leq J - 1$. The earlier a join, the greater its assigned weight. Next, given the annealing result matrix A, which is a RxJ matrix containing all variables *roj* for each relation r and join j, we determine the vector $\vec{c} = A\vec{w}$. The more early joins r is assigned to, the larger the value c_r, which hence numerically expresses the plausibility of r being joined earlier rather than later.

Finally, based on $\vec{c}$, we derive a rank vector $\vec{rk}$ that ranks all relations in accordance to $\vec{c}$, and hence can be used as a join order. In addition, for each join order, we always derive a *fallback solution*, by strictly traversing the query graph in accordance to $\vec{rk}$. The fallback solution hence deliberately excludes cross products. Thereby, we address inherent inaccuracies of the annealing process and our cost approximation method, which may sometimes lead to the inclusion of unfavorable cross products, which quickly yield large intermediate results. For these cases, the fallback solution provides an alternative solution, which ideally compares to solutions obtained by competing JO approaches without cross products.

Example 5.1. *Consider the annealing result matrix*

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

for a query joining four relations. It represents an invalid variable assignment, as relations 2 and 4 are not continuously present for all succeeding joins once initially joined by joins 1 and 2 respectively, while relation 3 has not been assigned a join at all. Using our readout method, we determine $\vec{c} = A\vec{w} = \begin{bmatrix} 6 & 4 & 0 & 2 \end{bmatrix}^T$. *Based on* $\vec{c}$, *we further arrive at the rank vector* $\vec{rk} = \begin{bmatrix} 1 & 2 & 4 & 3 \end{bmatrix}^T$. *Finally, we may directly use* $\vec{rk}$ *as a join order, and have hence achieved our goal of reading out a valid join order from an erroneous annealing result.*

6 EXPERIMENTAL ANALYSIS

Our novel native encoding allows JO on quantum-inspired HW. To evaluate the benefits, we next conduct an experimental analysis that compares our DA method against a variety of competing methods. We describe our experimental setup in Sec. 6.1, and analyse the scalability of our method for large solution spaces in Sec. 6.2.

6.1 Experimental Setup

When considering the overall query optimisation workflow, which includes steps such as query parsing and deriving statistical data, we are only concerned with the join ordering step. Hence, our problem input is given by a query graph, labeled with cardinalities, predicates and their selectivities, as derived by the preceding query optimisation steps. Similarly to Neumann and Radke [38], our experimental analysis considers query graphs extracted from various workloads, which we pass to all algorithms used for our experiments. We then evaluate the join order yielded by an algorithm based on intermediate result sizes, following the cost function c_{out}. Finally, costs are normalised to the best solution determined by any algorithm considered by our analysis.

Algorithms. Our goal is to show scalability benefits of our approach for large solution spaces. We compare our algorithm against competing methods that are known to scale up to at least moderate query sizes, and deliberately do not consider approaches known to offer very limited scalability, such as DP algorithms for super-polynomially growing search spaces.[5], which fail to obtain solutions for queries beyond 20 joins [53].

We consider the algorithm selection of Neumann and Radke [38] representative of the state-of-the-art. To ascertain fairness, we compare the DA, running our novel encoding, to algorithms operating on left-deep solution spaces. We also include baselines to analyse the effectiveness of our novel encoding and special-purpose DA HW, leading to the following selection:

- DPSizeLinear: *DP* method by Selinger *et al.* [43], yielding optimal solutions *without cross products* for *any query graph.*
- IKKBZ: *Polynomial-time heuristic* algorithm [18, 23], yielding solutions *without cross products* for *acyclic query graphs.*
- Minsel: *Greedy heuristic* [46], yielding solutions *without cross products* (CPs) for *any query graph.*
- MILP: *Special-purpose optimisation algorithm* [53], yielding solutions *with CPs* for *any query graph.*
- Genetic: *Genetic algorithm* [44], yielding solutions *with CPs* for *any query graph.*
- SA (novel encoding): *Simulated Annealing*, running our *novel encoding* on *classical HW*, yielding solutions *with CPs* for *any query graph.*
- DA (baseline): *Digital Annealing*, based on the *baseline approximation* [41], yielding solutions *with CPs* for *any query graph.*

Our set of algorithms features representatives over a wide range of methods, both established and novel for query optimisation. We include exhaustive search to obtain optimal solutions for restricted search spaces (DPSizeLinear), polynomial-time heuristics for swiftly obtaining solutions (IKKBZ), greedy heuristics (Minsel), metaheuristics (Genetic), special-purpose solvers with decades of maturing, running on conventional HW (MILP), and finally, our novel annealing method, tailored to fully exhaust the capabilities of highly optimised special-purpose HW (DA). As further baselines, we add SA on classical HW, running our novel encoding, to analyse the performance advantage of special-purpose DA HW, and an encoding with baseline cost approximation [41][6], run on the DA,

[5] For JO, such approaches include DP algorithms considering cross products.

[6] By augmenting the baseline encoding [41] with our novel encoding strategy for valid join trees, allowing the use of our novel readout, the encoding yields an *upper bound*

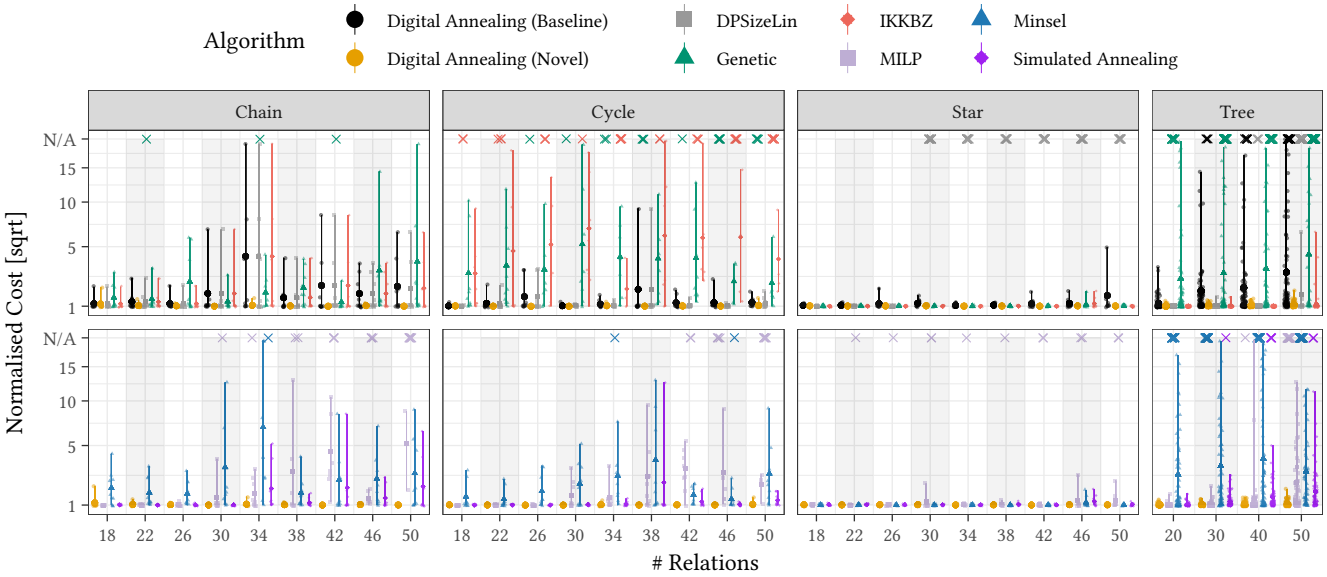

Figure 3: Normalised costs (relative to overall best solutions; square root scale) achieved by various JO approaches for chain, star, cycle and tree queries joining increasing numbers of relations. Point ranges depict average, minimal and maximal normalised costs obtained within a batch; individual solutions are given by small points. Cases where an algorithm does not produce a valid result after 60s, or results with normalised costs equal or greater 20, are denoted by crosses (slightly jittered horizontally and vertically to resolve outlier count) with cost N/A. Shaded areas group results for different input sizes. We split the comparison of our approach (orange circles, identical values across rows) against the state-of-the-art into two rows to avoid visual clutter.

to evaluate the advantage of our novel encoding. Below, we assess which approach is most capable for optimising large queries.

For DPSizeLinear, IKKBZ and Minsel, we use implementations by Neumann and Radke [38], kindly provided to us in executable form. We use the genetic algorithm implemented by Trummer [50], adjusted to use the cost function C_{out}. For MILP, we use the original Java code of Trummer and Koch [53], using the Gurobi optimiser (version 10.0.0), and reuse their approximation quality setup (they apply thresholds as powers of 10). We consider two additional approximation quality setups, with thresholds as powers of 2 and 100, which correspond to higher and lower approximation qualities, respectively. For each query, we report the best MILP result obtained over all configurations. All algorithms are set to time out after 60s.

Our JO-QUBO encodings, implemented in Python 3.8.16, are available in the reproduction package. We run the SA dwave-neal solver (version 0.6.0) on an octa-core AMD Ryzen 7 PRO 5850U CPU with 32 GB of DDR4 RAM, and use the second generation DA. Since optimisation time depends on the number of annealing steps, or sweeps, we conduct runs with varying amounts (up to 10^4 for SA and 10^8 for DA[7]), such that total optimisation time does not exceed 60s. For the remaining parameters, we use default values. We choose logarithmic thresholds based on exponentials of 2 as described in Sec. 4.2. During preliminary testing, we observed a dependence between the number of thresholds and QUBO penalty weight determination (see Sec. 4.3), where increasing the former complicates the latter. To reduce the load on the DA, we hence apply three threshold configurations with single thresholds 0.63, 2.55 and 5.11 respectively, and moreover one configuration without any thresholds, relying entirely on quadratic approximation. Even such minimal threshold configurations are sufficient to achieve substantial performance with our novel encoding, as we will show.

Like MILP, for each query, we report the best result derived over all configurations[8]. For each experiment and configuration, we conduct up to ten annealing runs[9], each yielding 100 shots, or solutions. Using our readout method as described in Sec. 5, we decode each annealing result into a valid join order. In addition, we obtain a fallback solution without cross products, by traversing the query graph based on the annealing result, as described in Sec. 5.

6.2 Evaluating Scalability

Common benchmarks for JO include TPC-H [48], TPC-DS [49], LDBC BI [2], SQLite [15] and the Join Order Benchmark [24]. Given their small or simple queries[10], their usefulness for assessing the scalability advantage of DA is limited (a detailed evaluation for these queries can be found in Appendix B, confirming the novel DA method performs well in all scenarios).

To assess the performance on larger queries, we need suitable loads. We consider ten batches of synthetic queries generated with the conventional method by Steinbrunn *et al.* [44], using code by

on the performance of the baseline encoding [41]. This allows a thorough performance comparison and moreover reduces variable requirements for the baseline encoding, such that queries of all considered sizes fit onto the DA.

[7]These represent upper bounds on orders of magnitude of annealing steps, as further increases exceed our time limit of 60s.

[8]A configuration consists of threshold values and annealing steps.

[9]For some queries, we conduct only a single annealing run, which is sufficient to obtain good join order with high probability.

[10]With exception of SQLite, queries included in these benchmarks join at most 18 relations, with median sizes substantially lower, whereas simply joining relations in accordance to the PK/FK structure obtains optimal SQLite solutions.

Trummer [50]. We generate 270 queries with up to 50 relations for the classical chain, star, and cycle query graphs. In addition, we evaluate our method on tree queries by Neumann and Radke [38], where we consider 400 queries joining up to 50 relations. All queries can be found in the reproduction package.

This captures many practically relevant and common scenarios with varying properties, which allows to assess the general utility of DA on JO, specifically (1) its capability to identify beneficial cross product opportunities, which occur more frequently in chain and cycle graphs than star graphs[11], (2) its ability to solve cyclic queries, which are problematic for some JO methods such as IKKBZ [18, 23], and finally (3), the DA's aptitude to obtain good solutions even for complex star and tree queries, a task known to be NP-complete when considering cross products [7, 8]. In contrast, the complexity of optimising chain and cycle queries with cross products is not fully understood [32]. However, given its factorial growth, their extremely large solution space is still sufficiently challenging to assess the scalability aptitude of JO algorithms, which becomes apparent in our experiments. For all queries, Fig. 3 on page 10 depicts the normalised cost of the obtained results. A more detailed analysis on execution time can be found in Appendix C.

6.2.1 DPSizeLinear. Using exhaustive search, DPSizeLinear guarantees optimality for a linear solution space without cross products. We observe that the algorithm manages to obtain such solutions after a few milliseconds for chain and cycle queries joining up to 50 relations. The query graph linearity and lack of cross product support restrict the amount of possible solutions for these queries, and enable DPSizeLinear to obtain solutions even for large queries.

Limits arise for tree or star graphs, whose structure engenders a large solution space even without cross products. For star graphs, DPSizeLinear initially obtains solutions within a few seconds. From 26 relations on, optimisation typically takes around 45s, and starting at 30 relations, no solution can be obtained before the 60s timeout. For tree queries, timeouts appear for queries joining 40 relations, and become frequent for 50 relations. While solutions are optimal without cross products, their exclusion severely degrades solution quality for chain and cycle queries: Normalised solution costs exceed competitors by up to a factor of almost 20.

6.2.2 IKKBZ. Unlike exhaustive search, IKKBZ heuristically determines linear JO solutions without cross products in polynomial time, enabling it to obtain good solutions even for large star queries within milliseconds. This excludes cyclic queries, where IKKBZ fails: While still obtaining solutions, their normalised costs often exceed those of other methods by orders of magnitude. Like DPSizeLinear, IKKBZ is unable to scale for large, general query graphs. Solution quality substantially suffers from excluded cross products.

6.2.3 Minsel. Similarly to IKKBZ and DPSizeLinear, Minsel considers a solution space without cross products. However, even for this restricted space, normalised costs for Minsel solutions often exceed those of competitors by orders of magnitude. We therefore observe an inconsistent performance that is characteristic for greedy algorithms, making the method inferior to most competitors.

[11] Once joining the central relation in star queries, which is sensible within the first few joins, cross products are no longer relevant. They only matter at the early stage. We can assume fewer cross product opportunities for star than for chain/cycle queries.

6.2.4 MILP. In contrast to the methods discussed so far, the MILP solver accounts for cross products, and explores an extremely large search space with factorial growth in the number of relations. Despite this challenge, the approach often obtains favourable cross product solutions for chain and cycle queries, which are substantially cheaper (up to a factor of almost 20) than join orders obtained by DPSizeLinear and IKKBZ. When a normalised (near-)optimal or solution (w.r.t. the best solution obtained by any algorithm) is obtained, run-time varies from a few milliseconds (simple cases) to over 20s (large queries). Our results also uncover a main drawback, as the MILP solver frequently fails to obtain a solution before the 60s timeout. The frequency of timeouts tends to increase as the number of relations grows. This illustrates the search space explosion that brings contemporary MILP solvers to their limits.

6.2.5 Genetic. Like MILP, the genetic algorithm considers a solution space with cross products, thereby often obtaining higher solution quality than IKKBZ or DPSizeLinear, in particular for chain and cycle queries. However, unlike MILP, the genetic solver guarantees to yield a join order after each optimisation step. Yet, solution quality varies greatly, often exceeding normalised costs of 20, making the genetic algorithm unsuitable for practical scenarios.

6.2.6 SA (Novel Encoding). SA on conventional HW (using our novel encoding) performs more robust compared to the genetic algorithm and MILP, while ensuring valid solutions. However, we observe cases where normalised costs exceed competitors by orders of magnitude. SA encounters runtime issues with an increasing number of relations: For default annealing time, a single batch of 100 solutions requires roughly 50s for 50 relations, almost reaching our 60s timeout. The large JO solution space hence remains challenging for SA on classical HW, even when using our novel encoding.

6.2.7 DA (Baseline Approximation). On the DA, obtaining a batch of 100 solutions for queries joining 50 relations takes roughly 5-7s, using the default 10^6 annealing steps. This illustrates the substantial runtime advantage of DA HW over SA on classical HW, by roughly an order of magnitude. We first analyse the encoding using baseline cost approximation following Schönberger *et al.* [41]. Yielding inconsistent performance, the method is frequently outperformed by competitors. We attribute this to the sparse set of single thresholds, unable to yield sufficient cost approximation following the baseline method [41]. The method hence often fails to obtain solutions containing beneficial cross products. However, increasing the amount of thresholds proves difficult on the second generation DA, making penalty weight tuning complex, as discussed above. The baseline method hence does not conform well to contemporary DA HW, due to complex parameter tuning and suboptimal performance.

6.2.8 DA (Novel Encoding). Finally, using our novel encoding, the DA addresses the drawbacks of competing approaches while retaining their strengths. Like MILP, the DA explores an extremely large search space with cross products, where it often obtains cheaper solutions compared to IKKBZ and DPSizeLinear. However, in stark contrast to MILP, which suffers from frequent timeouts, and the genetic algorithm, which frequently obtains very costly solutions, the DA handles the extreme size of the solution space gracefully, as it guarantees to yield the desired number of annealing solutions, each of which we translate into a valid join order. While this provides no

guarantees on the solution quality, our empirical results show that the DA, using our novel encoding, obtains normalised optimal or near-optimal solutions for *all* considered queries, *independently of the query graph shape and size*. Normalised optimal or near-optimal solutions are typically obtained within the first batch of 100 annealing runs. Depending on query size, a batch requires predictable optimisation time of 5-7s. For JO, the DA is roughly an order of magnitude faster than SA on classical HW.

Using our novel encoding, the DA provides the most consistent performance out of all considered JO algorithms, typically obtaining normalised optimal or near-optimal solutions within seconds, even with a sparse set of thresholds. To further increase solution quality, one may apply a broader selection of thresholds, which requires additional annealing runs. To address the increased optimisation time, the number of individual solutions sampled for single runs may be decreased. We recommend thresholds based on powers of 2 to enable our variable-efficient binary encoding of Sec. 4.2.

Our results, though already substantial, only give *lower* performance bounds, as we use single cost approximation thresholds. Newer DA HW offers advanced methods for penalty weight tuning [12]. This may ease using large amounts of thresholds, thus reducing the impact of individual thresholds, and increasing solution quality.

7 RELATED WORK

The JO problem is among the most extensively studied query optimisation problems [24, 32, 37, 38, 53, 57]. Exhaustive search methods such as DP [30, 33, 34, 43, 54] provide optimal solutions within their solution spaces, yet face limitations when scaling up problem dimensions. Heuristic methods [6, 17, 20, 44, 46, 47, 52] are more suitable for large queries, yet provide no guarantees on solution quality. This category contains the method discussed in this paper. Query optimization methods based on machine learning [21, 25, 26] have become popular in the past years. Corresponding work typically focuses on optimising smaller queries more reliably (*e.g.*, by learning data correlations [21]), rather than addressing challenges due to particularly large search spaces. Recent work in this domain [26] exploits traditional optimisers as a sub-function and could therefore benefit from faster join ordering methods as well.

Approaches like MILP have been applied on a variety of DB optimisation problems: Multiple query optimisation [10], materialised view design [56], or index selection [40]. Ref. [53] gives a MILP encoding for JO; yet, large JO solution spaces remain problematic for highly optimised solvers on conventional HW.

Rather than conventional HW, we consider the Fujitsu DA, which applies quantum-inspired extensions to classical SA, to exploit its high parallelisation capabilities enabled by special-purpose HW. Thereby, the DA can achieve speedups of two orders of magnitude over classical SA for dense QUBO problems, as shown by Aramon *et al.* [3]. Matsubara *et al.* [28] provide an overview on DA for a variety of combinatorial optimisation problems. For the MinCut and MaxCut problems, the DA outperforms dedicated solvers in both, solution quality and runtime performance, achieving speedups over conventional methods by orders of magnitude. Similar improvements have recently been achieved for problems in various domains, including sensor placement [19], optical network modernisation [45], market graph clustering [16] and segment routing [11].

Our DA method follows a trend within DB research that recognises limitations of general-purpose systems and addresses them using special-purpose HW. Related approaches include, for instance, an energy-efficient stream join for IoT [31], or resource-efficient FPGA modules for DB operators [35], among other research [12, 59].

QUBO encodings have been proposed for different DB problems, such as DB transaction scheduling [4, 5, 14] or multiple query optimisation [51]. However, these problems are unrelated to JO. Their problem-specific encodings cannot be reused for JO, and have so far only been applied to small-scale problems with quantum annealing. They presently do not scale, given current QPU HW limitations. In contrast, we demonstrate scalability for JO problems with extremely large search spaces, using highly optimised quantum-inspired HW.

8 DISCUSSION AND CONCLUSION

The shift from general-purpose HW to optimised special-purpose systems, motivated by the prospect of higher energy efficiency and computational throughput, is an established trend within many fields, including database research. To exhaust the full potential of these systems, a comprehensive perspective encompassing both, novel problem encodings (SW) and architectural aspects (HW) is required. Driven by such co-design considerations, we are prompted to re-evaluate existing optimisation methods, designed and matured for conventional systems, tailoring them to special-purpose HW.

The potential of such systems remains unexplored for many database problems, which motivates us to evaluate their aptness in the domain, starting with its most fundamental problems. We have analysed the potential of quantum-inspired special-purpose HW on the join ordering problem, which (1) is one of the most fundamental query optimisation problems, and (2) features an extremely large solution space, making it a challenging and suitable candidate to assess the aptitude of special-purpose HW.

Given the large JO solution space, obtaining exact optimal solutions with exhaustive search becomes infeasible beyond small-sized queries. While many established methods consider a reduced search space for feasible exploration, the size reduction can severely degrade solution quality, as shown by our results. Finding ways to handle large solution spaces remains desirable, and previous attempts include the use of MILP solvers. Our results show their limitations, as MILP optimisation frequently fails as queries grow.

We solve queries on the quantum-inspired Fujitsu DA, using a novel, native JO-QUBO encoding tailored to the DA's architecture. Unlike MILP, our DA approach guarantees, within predictable optimisation times, to yield the desired number of solutions even for large queries joining 50 relations, which far exceeds typical query dimensions, including complex join orders unobtainable within the restricted search space considered by competing approaches. Our results demonstrate the aptitude of special-purpose HW to achieve scalability even for extremely large JO solution spaces, and prompt their further evaluation on yet unconsidered problems.

Acknowledgements. MS and WM were supported by the German Federal Ministry of Education and Research (BMBF), funding program "Quantum Technologies—from Basic Research to Market", grant number 13N15647. WM acknowledges support by the High-Tech Agenda Bavaria.

[12] Generations 3 and 4 determine weights by separating validity and cost terms.

REFERENCES

[1] 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

[2] Renzo Angles, Peter Boncz, Josep-Lluis Larriba-Pey, Irini Fundulaki, Thomas Neumann, Orri Erling, Peter Neubauer, Norbert Martínez-Bazan, Venelin Kotsev, and Ioan Toma. 2014. The Linked Data Benchmark Council: A graph and RDF industry benchmarking effort. *ACM SIGMOD Record* 43 (05 2014), 27–31. https://doi.org/10.1145/2627692.2627697

[3] Maliheh Aramon, Gili Rosenberg, Elisabetta Valiante, Toshiyuki Miyazawa, Hirotaka Tamura, and Helmut G. Katzgraber. 2019. Physics-Inspired Optimization for Quadratic Unconstrained Problems Using a Digital Annealer. *Frontiers in Physics* 7 (2019). https://doi.org/10.3389/fphy.2019.00048

[4] 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

[5] 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.

[6] Nicolas Bruno, César Galindo-Legaria, and Milind Joshi. 2010. Polynomial heuristics for query optimization. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. 589–600. https://doi.org/10.1109/ICDE.2010.5447916

[7] S. Chatterji, S. S. K. Evani, S. Ganguly, and M. D. Yemmanuru. 2002. On the Complexity of Approximate Query Optimization. In *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (Madison, Wisconsin) *(PODS '02)*. Association for Computing Machinery, New York, NY, USA, 282–292. https://doi.org/10.1145/543613.543650

[8] 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.

[9] Nicolas Dieu, Adrian Dragusanu, Françoise Fabret, François Llirbat, and Eric Simon. 2009. 1,000 Tables Under the From. *PVLDB* 2, 2 (2009), 1450–1461. https://doi.org/10.14778/1687553.1687572

[10] Tansel Dokeroglu, Murat Ali Bayır, and Ahmet Cosar. 2014. Integer Linear Programming Solution for the Multiple Query Optimization Problem. In *Information Sciences and Systems 2014*, Tadeusz Czachórski, Erol Gelenbe, and Ricardo Lent (Eds.). Springer International Publishing, Cham, 51–60.

[11] Sebastian Engel, Christian Münch, Fritz Schinkel, Oliver Holschke, Marc Geitz, and Timmy Schüller. 2022. Segment Routing with Digital Annealing. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. 1–9. https://doi.org/10.1109/NOMS54207.2022.9789782

[12] Johannes Fett, Annett Ungethüm, Dirk Habich, and Wolfgang Lehner. 2021. The Case for SIMDified Analytical Query Processing on GPUs. In *Proceedings of the 17th International Workshop on Data Management on New Hardware (DaMoN 2021)* (Virtual Event, China) *(DAMON'21)*. Association for Computing Machinery, New York, NY, USA, Article 14, 5 pages. https://doi.org/10.1145/3465998.3466015

[13] Fujitsu Limited. 2023. Fujitsu Digital Annealer - solving large-scale combinatorial optimization problems instantly. https://www.fujitsu.com/emeia/services/business-services/digital-annealer/what-is-digital-annealer/

[14] 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

[15] Richard D Hipp. 2020. SQLite. https://www.sqlite.org/index.html

[16] Seo Woo Hong, Pierre Miasnikof, Roy Kwon, and Yuri Lawryshyn. 2021. Market Graph Clustering via QUBO and Digital Annealing. *Journal of Risk and Financial Management* 14, 1 (2021). https://doi.org/10.3390/jrfm14010034

[17] Jorng-Tzong Horng, Cheng-Yan Kao, and Baw-Jhiune Liu. 1994. A genetic algorithm for database query optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. 350–355 vol.1. https://doi.org/10.1109/ICEC.1994.349926

[18] Toshihide Ibaraki and Tiko Kameda. 1984. On the Optimal Nesting Order for Computing N-Relational Joins. *ACM Trans. Database Syst.* 9, 3 (sep 1984), 482–502. https://doi.org/10.1145/1270.1498

[19] Tomoki Inoue, Tsubasa Ikami, Yasuhiro Egami, Hiroki Nagai, Yasuo Naganuma, Koichi Kimura, and Yu Matsuda. 2023. Data-driven optimal sensor placement for high-dimensional system using annealing machine. *Mechanical Systems and Signal Processing* 188 (2023), 109957. https://doi.org/10.1016/j.ymssp.2022.109957

[20] Y. E. Ioannidis and Younkyung Kang. 1990. Randomized Algorithms for Optimizing Large Join Queries. *SIGMOD Rec.* 19, 2, 312–321. https://doi.org/10.1145/93605.98740

[21] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. 2018. Learned cardinalities: estimating correlated joins with deep learning. In *CIDR*. 1–8. arXiv:1809.00677 http://arxiv.org/abs/1809.00677

[22] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by Simulated Annealing. *Science* 220, 4598 (1983), 671–680. https://doi.org/10.1126/science.220.4598.671 arXiv:https://www.science.org/doi/pdf/10.1126/science.220.4598.671

[23] Ravi Krishnamurthy, Haran Boral, and Carlo Zaniolo. 1986. Optimization of Nonrecursive Queries. In *Proceedings of the 12th International Conference on Very Large Data Bases (VLDB '86)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 128–137.

[24] 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.

[25] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2018. Neo: A Learned query optimizer. *PVLDB* 12, 11 (2018), 1705–1718. https://doi.org/10.14778/3342263.3342644 arXiv:1904.03711

[26] Tim Marcus, Ryan and Negi, Parimarjan and Mao, Hongzi and Tatbul, Nesime and Alizadeh, Mohammad and Kraska. 2022. Bao: Making Learned Query Optimization Practical. In *ACM SIGMOD Record*, Vol. 51. 6–13. https://doi.org/10.1145/3542700.3542702

[27] 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.

[28] Satoshi Matsubara, Motomu Takatsu, Toshiyuki Miyazawa, Takayuki Shibasaki, Yasuhiro Watanabe, Kazuya Takemoto, and Hirotaka Tamura. 2020. Digital Annealer for High-Speed Solving of Combinatorial optimization Problems and Its Applications. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 667–672. https://doi.org/10.1109/ASP-DAC47756.2020.9045100

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

[30] 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

[31] Adrian Michalke, Philipp M. Grulich, Clemens Lutz, Steffen Zeuch, and Volker Markl. 2021. An Energy-Efficient Stream Join for the Internet of Things. In *Proceedings of the 17th International Workshop on Data Management on New Hardware (DaMoN 2021)* (Virtual Event, China) *(DAMON'21)*. Association for Computing Machinery, New York, NY, USA, Article 8, 6 pages. https://doi.org/10.1145/3465998.3466005

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

[33] 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.

[34] Guido Moerkotte and Thomas Neumann. 2008. Dynamic Programming Strikes Back. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (Vancouver, Canada) *(SIGMOD '08)*. Association for Computing Machinery, New York, NY, USA, 539–552. https://doi.org/10.1145/1376616.1376672

[35] Mehdi Moghaddamfar, Christian Färber, Wolfgang Lehner, Norman May, and Akash Kumar. 2021. Resource-Efficient Database Query Processing on FPGAs. In *Proceedings of the 17th International Workshop on Data Management on New Hardware (DaMoN 2021)* (Virtual Event, China) *(DAMON'21)*. Association for Computing Machinery, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3465998.3466006

[36] Hiroshi Nakayama, Junpei Koyama, Noboru Yoneoka, and Toshiyuki Miyazawa. 2021. Description: Third Generation Digital Annealer Technology. https://www.fujitsu.com/global/documents/about/research/techintro/3rd-g-da_en.pdf

[37] 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

[38] 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

[39] 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.

[40] Stratos Papadomanolakis and Anastassia Ailamaki. 2007. An Integer Linear Programming Approach to Database Design. In *2007 IEEE 23rd International Conference on Data Engineering Workshop*. 442–449. https://doi.org/10.1109/ICDEW.2007.4401027

[41] Manuel Schönberger, Stefanie Scherzinger, and Wolfgang Mauerer. 2023. Ready to Leap (by Co-Design)? Join Order Optimisation on Quantum Hardware. In *Proceedings of the 2023 ACM International Conference on Management of Data*. ACM, Seattle, WA, USA.

[42] Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. 2023. Quantum Optimisation of General Join Trees. In *Joint Workshops at 49th International Conference on Very Large Data Bases (VLDBW'23) — International Workshop on Quantum Data Science and Management (QDSM '23)*.

[43] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. 1979. Access Path Selection in a Relational Database Management System. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data* (Boston, Massachusetts) *(SIGMOD '79)*. Association for Computing Machinery, New York, NY, USA, 23–34. https://doi.org/10.1145/582095.582099

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

[45] Masahiko Sugimura, Mikinori Kobayashi, Hidetoshi Matsumura, Xi Wang, and Paparao Palacharla. 2022. Accelerate Optical Network Modernization through Quantum-inspired Digital Annealing. In *2022 European Conference on Optical Communication (ECOC)*. 1–4.

[46] A. Swami. 1989. Optimization of Large Join Queries: Combining Heuristics and Combinatorial Techniques. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data* (Portland, Oregon, USA) *(SIGMOD '89)*. Association for Computing Machinery, New York, NY, USA, 367–376. https://doi.org/10.1145/67544.66961

[47] Arun Swami and Anoop Gupta. 1988. Optimization of Large Join Queries. *SIGMOD Rec.* 17, 3, 8–17. https://doi.org/10.1145/971701.50203

[48] Transaction Processing Performance Council . 2023. TPC Benchmark H. https://www.tpc.org/

[49] Transaction Processing Performance Council. 2023. TPC Benchmark DS. https://www.tpc.org/

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

[51] 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.

[52] Immanuel Trummer and Christoph Koch. 2016. Parallelizing Query Optimization on Shared-Nothing Architectures. *Proc. VLDB Endow.* 9, 9 (may 2016), 660–671. https://doi.org/10.14778/2947618.2947622

[53] 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.

[54] Bennet Vance and David Maier. 1996. Rapid Bushy Join-Order Optimization with Cartesian Products. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* (Montreal, Quebec, Canada) *(SIGMOD '96)*. Association for Computing Machinery, New York, NY, USA, 35–46. https://doi.org/10.1145/233269.233317

[55] A. Adrian Vogelsgesang, M. Michael Haubenschild, J. Jan Finis, A. Alfons Kemper, V. Viktor Leis, T. Tobias Muehlbauer, T. Thomas Neumann, and M. Manuel Then. 2018. Get real: How benchmarks Fail to Represent the Real World. In *DBTest*. https://doi.org/10.1145/3209950.3209952

[56] Jian Yang, Kamalakar Karlapalem, and Qing Li. 1997. Algorithms for Materialized View Design in Data Warehousing Environment. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB '97)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 136–145.

[57] 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

[58] 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.

[59] Qitian Zeng, Kyle C. Hale, and Boris Glavic. 2021. Playing Fetch with CAT: Composing Cache Partitioning and Prefetching for Task-Based Query Processing. In *Proceedings of the 17th International Workshop on Data Management on New Hardware (DaMoN 2021)* (Virtual Event, China) *(DAMON'21)*. Association for Computing Machinery, New York, NY, USA, Article 15, 5 pages. https://doi.org/10.1145/3465998.3466016

A MORE ON RESOURCE SCALING

In Sec. 4.3, we demonstrated the resource efficiency of our novel encoding as compared to the baseline encoding proposed by Schönberger *et al.* [41] for *mandatory variables*, required to encode queries independently of threshold variables used for the cost approximation. For a more comprehensive comparison, let us now moreover consider *approximation variables*, required to encode thresholds for the cost approximation. The amount of thresholds, and hence the number of required approximation variables, depends on the desired approximation quality. Specifically, a higher approximation quality requires a larger amount of approximation variables.

Following Schönberger *et al.* [41] for the baseline encoding, and Sec. 4 for our novel encoding, the respective numbers of approximation variables $n_{baseline}$ and n_{novel} are given by

$$n_{baseline} = T(J-1) + T\sum_{j=1}^{J-1}\left(\left\lfloor \log_2\left(\frac{c_{j_{\max}}}{\omega}\right)\right\rfloor + 1\right),$$

$$n_{novel} = \sum_{j=1}^{J-1}\sum_{t=1}^{T}\left(\left\lfloor \log_2\left(\frac{\log_{10}(\theta_t)}{\omega}\right)\right\rfloor + 1\right),$$

for J joins and T threshold values, where ω denotes the precision for discretising continuous variables. Importantly, the number of cost variables required for the baseline method depends on the term $c_{j_{\max}}$, which denotes the maximum logarithmic cardinality possible for a join j, whereas for our new approach, it depends on the logarithmic threshold $\log(\theta_t)$.

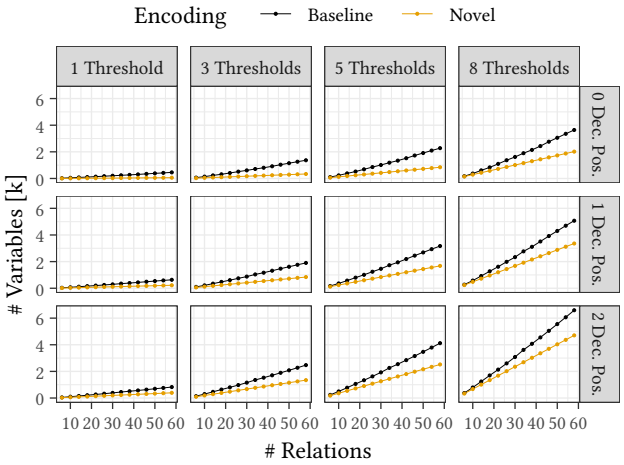

Figure 4: Approximation variable scaling for cycle queries with up to 58 relations and various numbers of decimal positions (D) and numbers of threshold values (T). We compare the baseline cost approximation method applied by Schönberger *et al.* [41] against our novel quadratic method.

Fig. 4 depicts concrete amounts of approximation variables for cycle queries with up to 58 relations. We compare multiple approximation configurations, ranging between one and eight thresholds and up to two decimal positions. Note that we choose logarithmic thresholds based on powers of 2 in accordance to the number of decimals, as discussed in Sec. 4.2. For each class of decimal positions, we select the first eight thresholds greater than 1.

Some of the final logarithmic thresholds exceed values of 200, corresponding to extremely large actual thresholds. While these no longer offer any practical use for actual queries, we still include them to analyse variable scaling. In contrast, the variable scaling for the baseline approach depends on the maximum logarithmic cardinalities for joins, which can assume very large values for practical queries, in particular when scaling up problem dimensions.

Even with extremely large thresholds, our novel approach requires significantly less variables than the baseline. The difference is even more pronounced for thresholds of practically useful sizes (*i.e.*, five or less thresholds), where our new method often saves more than 50% of approximation variables. Our novel encoding therefore not only demonstrates higher efficiency in encoding mandatory variables, but moreover drastically reduces resource requirements for the cost approximation.

B PERFORMANCE ON STANDARD BENCHMARKS

In Sec. 6, we empirically demonstrated the scalability and robustness of our novel DA method on large, synthetic queries. In the following, we moreover analyse its aptitude for commonly applied JO benchmarks, including TPC-H [48], TPC-DS [49], LDBC BI [2], SQLite [15] and the Join Order Benchmark (JOB) [24]. We use query graphs extracted by Neumann and Radke [38], where we consider those queries conforming to our JO model outlined in Sec. 3 (queries containing elements beyond this model, *e.g.*, outer joins, are not supported by our method). We moreover exclude queries joining merely two relations, as these are trivial for our JO model, which does not discriminate between the two possible solutions.

With the exception of SQLite, queries for these benchmarks are very limited in size, joining 18 relations at most, with median sizes substantially smaller. In contrast, SQLite features larger queries joining up to 64 relations, where we consider queries with up to 60 relations for our analysis, as larger sizes are unsupported by the DA when using the baseline encoding. Despite the larger sizes, the complexity of SQLite queries is rather limited, as simply joining relations in accordance to the PK/FK structure yields optimal join orders. While these standard benchmarks hence provide only limited insights on scalability, their study complements our empirical analysis for synthetic queries, by covering a plethora of varying scenarios.

Algorithms. For JO methods considering a solution space without cross products, which include DPSizeLinear, IKKBZ and Minsel, we restrict our analysis to DPSizeLinear, which does not encounter scalability issues for the benchmark queries (as opposed to our synthetic queries), and obtains optimal results within such a restricted search space. Thereby, DPSizeLinear provides an upper bound on solution quality achievable by IKKBZ and Minsel. We further include all JO methods that consider an extended solution space with cross products, as analysed in Sec. 6 for synthetic queries, including MILP, the genetic algorithm, SA (using our novel encoding), and DA (using both, an encoding with baseline cost approximation following Schönberger *et al.* [41], and our novel encoding).

Note that we cannot rely on the DPSizeLinear implementation by Neumann and Radke [38] for the benchmark queries, as their

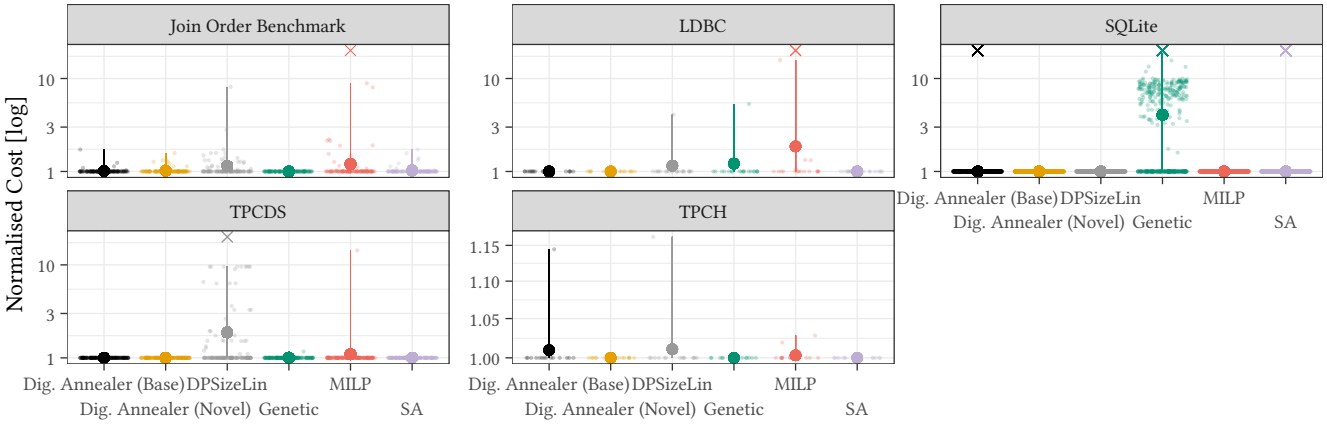

Figure 5: Normalised costs (relative to overall best solutions; square root scale) achieved by various JO approaches for benchmark queries. Point ranges depict average, minimal and maximal normalised costs obtained within a batch; individual solutions are given by small points. Cases where an algorithm does not produce a valid result after 60s, or results with normalised costs equal or greater 20, are denoted by crosses (slightly jittered horizontally and vertically to resolve outlier count) with cost N/A. Shaded areas group results for different input sizes.

cost calculation differs for certain scenarios found in benchmark queries. Specifically, if cyclic queries are encountered, their cardinality estimation is based on the minimum spanning tree for joins w.r.t. selectivities (this scenario is neither encountered for our synthetic chain, star and tree queries, which do not contain cycles, nor our cycle queries, where the cycle is only created for the final join, which is not part of our cost calculation). To ascertain identical cost calculations, required for a fair comparison, we therefore use our own DPSizeLinear implementation, written in Python and provided in the reproduction package, which calculates costs following the cost function C_{out}, as described in Sec. 3.

Implementations and configurations for the remaining algorithms are as described in Sec. 6.1. Fig. 5 depicts normalised solution costs for all benchmark queries obtained by each algorithm.

Results. While DPSizeLinear performs well in many scenarios, we observe frequent cases where the method obtains solutions substantially more costly than competitors. Such scenarios occur most prominently for the TPC-DS benchmark, and moreover for LDBC and JOB queries, where cheaper solutions can be obtained by considering cross products, which are unaccounted for by DPSizeLinear.

While this demonstrates the benefits of exploring an extended solution space with cross products, the higher exploration complexity is challenging. For MILP, this manifests in costly solutions or timeouts encountered for a small selection of LDBC, JOB and TPC-DS queries. For the genetic algorithm, the increased exploration complexity is most prominently featured in results for SQLite queries, where optimal solutions are obtained by joining relations strictly in accordance to their PK/FK structure. The genetic algorithm fails to achieve the required level of precision beyond small-sized queries, resulting in large normalised costs for queries of moderate and larger sizes.

Results for large SQLite queries moreover reveal runtime limits encountered by algorithms like simulated annealing, which fails to obtain a single batch of solutions within our time limit of 60s once queries join roughly 55 relations. In contrast, as discussed for the synthetic queries in Sec. 6, the DA achieves a comparative speedup by roughly an order of magnitude, and hence deals with such scenarios gracefully. While both, our novel encoding and the baseline encoding achieve similar results for the benchmark queries, our analysis for larger, more complex synthetic queries, conducted in Sec. 6, revealed the limitations of the baseline.

C EXTENDED RUNTIME ANALYSIS

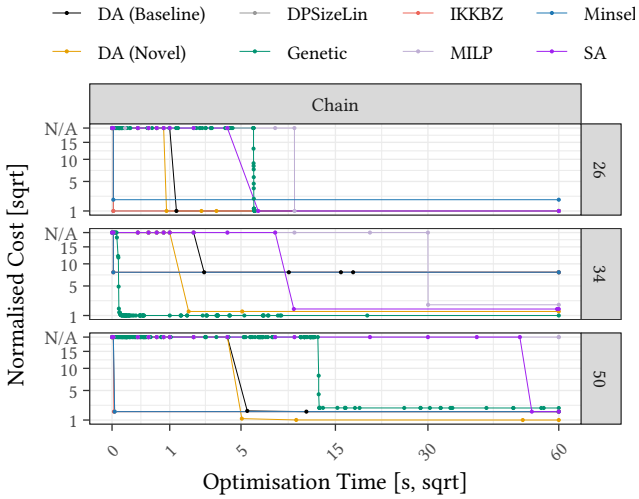

Figure 6: Time evolution of normalised costs with increasing computational budget (time) for chain queries. We show typical runs for queries of different sizes.

In Sec. 6, we demonstrated the robustness of our DA method against the extremely large size of the JO solution space containing

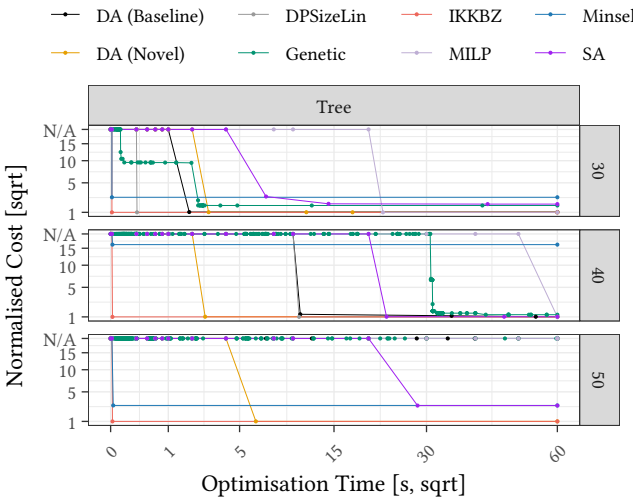

Figure 9: Time evolution of normalised costs with increasing computational budget (time) for tree queries. We show typical runs for queries of different sizes.

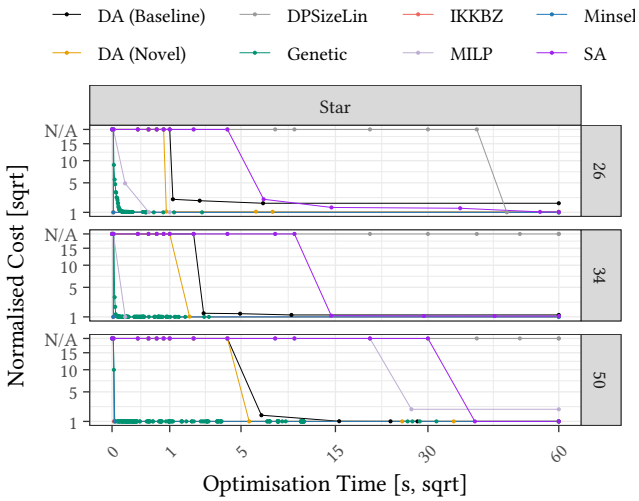

Figure 8: Time evolution of normalised costs with increasing computational budget (time) for star queries. We show typical runs for queries of different sizes.

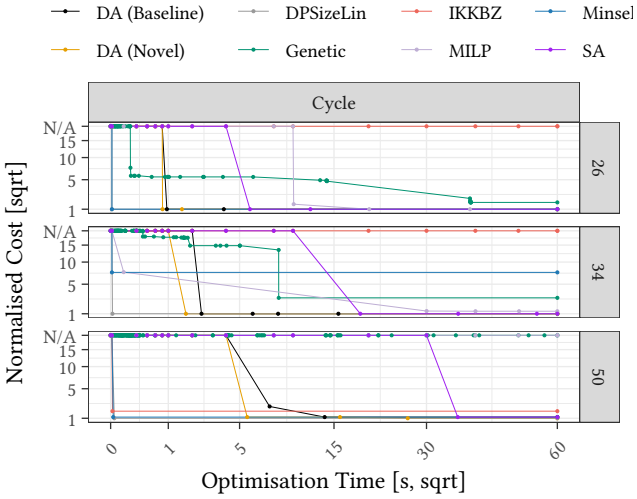

Figure 7: Time evolution of normalised costs with increasing computational budget (time) for cycle queries. We show typical runs for queries of different sizes.

cross products, allowing it to avoid the scalability issues of competing methods and to obtain cheaper plans in comparison to methods for more restricted solution spaces. In Sec. 6, our analysis thereby primarily focused on normalised solution cost. In the following, we discuss the runtime properties of our method in greater detail.

Figures 6, 7, 8 and 9 depict the time evolution of normalised costs for chain, cycle, star and tree queries respectively, as derived by the algorithms considered by our analysis. With the exception of DPSizeLinear, which is subject to scalability issues depending on the query graph structure, JO methods for a restricted solution space without cross products, such as IKKBZ and Minsel, require at most a few milliseconds to determine plans. However, their costs can exceed those of plans obtained within a more extensive solution space by orders of magnitude, as discussed in Sec. 6. Such methods include the genetic algorithm, MILP, SA, and DA.

While often benefitting from cheaper plans through cross products, both, MILP and the genetic algorithm, frequently require up to 20s to obtain normalised optimal or near-optimal solutions. In addition, MILP suffers from frequent timeouts, whereas solution costs for the genetic algorithm often exceed normalised optimal costs by a factor of 20 and beyond.

In contrast, DA, running our novel JO-QUBO encoding, typically requires merely one batch of 100 shots to obtain normalised optimal or near-optimal solutions. Even for large queries joining 50 relations, a single batch requires roughly 5-6s of optimisation time, at the default amount of 6 million annealing steps. Thereby, DA moreover beats SA on classical hardware, both in terms of solution quality and runtime, where a single batch of 100 solutions at the default amount of annealing iterations requires up to 50s, exceeding DA times by an order of magnitude. This demonstrates the performance advantage of special-purpose DA devices over classical HW.