Constrained Quadratic Model for Optimizing Join Orders

Pranshi Saxena
Information Sciences Institute
University of Southern California
Los Angeles, California, USA
pranshis@usc.edu

Ibrahim Sabek
Department of Computer Science
University of Southern California
Los Angeles, California, USA
sabek@usc.edu

Federico Spedalieri
Information Sciences Institute
University of Southern California
Los Angeles, California, USA
fspedali@isi.edu

ABSTRACT

We present a quantum-based approach for the optimization of join orders in database applications. Our approach relies on a hybrid framework, where classical heuristics are combined with a quantum processor to accelerate the search over the set of solutions to a constrained problem. By taking advantage of a previously introduced formulation of the join order optimization as a Quadratic Unconstrained Binary Optimization (QUBO) problem, we implement it using the Constrained Quadratic Model (CQM), a hybrid classical-quantum solver that interfaces classical heuristics with D-Wave's quantum annealer. We show that even a generic implementation of this classical-quantum hybrid framework produces competitive results for the join order problem, suggesting that better-tailored hybrid solvers could produce a computational advantage.

KEYWORDS

Join Order, Query Optimization, Quantum Computing, Quadratic Unconstrained Binary Optimization, Constrained Quadratic Model

ACM Reference Format:
Pranshi Saxena, Ibrahim Sabek, and Federico Spedalieri. 2024. Constrained Quadratic Model for Optimizing Join Orders. In *Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications (Q-Data '24), June 09–15, 2024, Santiago, AA, Chile.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3665225.3665447

1 INTRODUCTION

The join order (JO) optimization problem [25] is pivotal in any query optimizer, where the goal is to determine the most efficient sequence for joining multiple relations in a database join query from among all possible combinations of sequences. Due to its huge search space, JO is known to be an NP-hard combinatorial optimization problem where obtaining sufficiently good solutions becomes more challenging as the number of relations increases. For small join queries, dynamic programming (DP) approaches can deliver optimal solutions. However, for large join queries, DP approaches fail due to the exponential increase in their computational demands. In this case, pure classical computation solutions have to rely on heuristics that involve quality-efficiency tradeoffs to reduce the search space and come up with a solution in a reasonable time.

Q-Data '24, June 09–15, 2024, Santiago, AA, Chile
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0553-3/24/06
https://doi.org/10.1145/3665225.3665447

Meanwhile, solving combinatorial optimization problems, e.g., JO optimization, has been one of the main applications of quantum computing devices due to their ability to process a vast number of potential solutions at once rather than exploring each possibility sequentially as classical computers do. One of the first such devices, the D-Wave quantum annealer [16], aimed at exploiting quantum mechanical properties to find the ground state of the Ising model [20] (a representation of a set of interacting magnets). Since the Ising model is NP-hard [1], other combinatorial optimization problems could be mapped into and solved using the quantum annealer. Quadratic Unconstrained Binary Optimization (QUBO) [30] is an example of the problems that can be directly obtained from the Ising model by a simple remapping of the optimization variables (from $s_i = \{+1, -1\}$ to binary variables $x_i = \{0, 1\}$).

Although combinatorial optimization problems can be translated into Ising models and programmed into quantum annealers, the capabilities of the physical hardware impose restrictions that limit the range of problems that can be effectively implemented. A quantum annealer like the D-Wave processor is built by laying out superconducting loops on a 2-dimensional architecture. Each loop is associated with what is known as a quantum bit, i.e., qubit. Implementing the Ising model requires that we control interactions between pairs of qubits. However, in this superconducting architecture, only qubits that are geometrically close to each other can be made to interact. This results in a very sparse connectivity graph, with each qubit interacting only with a constant number of other qubits (currently around 15).

The limitations imposed by quantum hardware constraints result in specific problems not being suited for direct implementation on a quantum annealer. For example, *constraints* on the optimization variables cannot be implemented natively. However, there are some well-known tricks to get around some of these issues, like converting constraints into penalty terms that are added to the cost function and mapping a single variable into a chain of qubits to increase the effective degree of the connectivity graph. These techniques aim at replacing the original problem with one that can be implemented in the quantum annealer, with the guarantee that the optimal solution is the same for both. However, to make the mapping work, the above-mentioned techniques usually introduce other issues of their own (e.g., limiting the size of the problem to be solved) and may sometimes hinder the overall performance. It is also worth mentioning that quantum computers operating using the quantum circuit model can also be used to solve combinatorial optimization problems [11]. However, its solutions either require efficient quantum error correction algorithms, which are hard to achieve with the currently available NISQ (Noisy, Intermediate

Scale, Quantum) devices, or suffer from a very time-consuming procedure when optimizing parameters [2], which is not well suited for applications that need good solutions to be produced quickly.

A different approach is based on using a hybrid quantum-classical approach, where a classical algorithm pre-processes the problem description and iteratively generates QUBO instances that can be implemented natively in the quantum annealer. In this paper, we present such an approach using a novel tool available on the D-Wave API, referred to as the Constrained Quadratic Model (CQM) [14], that avoids the direct mapping of constraints. Even though this tool does not provide details about its inner workings, it allows us to show a baseline performance for classical-quantum hybrid approaches. This is (to our knowledge) the first application of this tool to the JO problem. Our results show that the CQM classical-quantum hybrid solver can reduce the cost of the best join order found by the most recent state-of-the-art QUBO-based JO encoding technique [29]. The average improvement can be between $\sim 3\%$ and $\sim 6\%$ for synthetic benchmarks and up to 20% for the JOB benchmark. These preliminary results show the potential of classical-quantum hybrid approaches and call for the development of more tailored implementations that could further improve performance.

2 BACKGROUND AND FUNDAMENTALS

2.1 Quantum Annealing (QA)

Quantum annealing (QA) has been proposed as a novel heuristic for combinatorial optimization problems [17], and it is similar to Simulated Annealing (SA). In SA, the configuration space is explored by randomly proposing changes to the state configuration: if the new value of the cause function is decreased, the proposed change is accepted; if the cost function increases, the change is accepted with some probability. In QA, these changes are applied on a quantum superposition of the configurations, allowing for a massively parallel search of the configuration space. This allows phenomena like quantum tunneling to facilitate the exploration, potentially avoiding getting stuck in local minima.

In QA, a quadratic cost function over binary variables is encoded in the interactions between two-level quantum mechanical systems, i.e., qubits, in a way that the solution to the problem of interest is the configuration that minimizes the energy of this interacting system. We can initialize the physical system in a state that is a quantum superposition of all possible configurations, and then we proceed to change the interaction strengths between the qubits slowly. This annealing process (if performed slowly enough) transforms the initial superposition into the state that minimizes the system's energy, which, in turn, encodes the solution to the combinatorial optimization problem.

The D-Wave quantum annealer [16] implements this model using superconducting technology, where small superconducting loops represent individual qubits: current circulating around these loops in both directions encode the 0 and 1 states of the qubit. For two qubits that are close to each other, a programmable interaction can be implemented between them, as well as local biases on the individual qubits. With these elements, an interacting system can be constructed whose energy takes the form

$$E(\vec{s}) = \sum_{ij} J_{ij}\, s_i\, s_j + \sum_i h_i\, s_i \tag{1}$$

where $\vec{s} = (s_1, \ldots, s_n)$ is a binary vector representing the system configuration, and (h_i, J_{ij}) are parameters representing the local biases on each qubit, and the interaction strengths between them.

One of the main limitations of quantum annealers is the restriction imposed on the available interactions due to the geometrical constraints imposed by the actual physical layout of the qubits. Ideally, we would like all qubits to be able to interact with every other qubit. However, to make two qubits interact, they must be close together, and when we lay out the physical superconducting loops on a two-dimensional surface, we see that this requirement is impossible to fulfill. Hence, each qubit can only interact with a fixed number of other qubits, resulting in a sparse connectivity graph. For the D-Wave Advantage system [34] that we use (that uses the Pegasus architecture [8]), the maximum degree of this graph is 15. Even though there are techniques to get around this issue by mapping a single variable to a chain of physical qubits that has a larger effective connectivity degree, we pay a price in the size of problems that can be implemented (since the total number of qubits in the processor is fixed). Encoding other problem features like constraints further reduces the number of qubits available to encode variables. However, as we shall see next, there are other ways of exploiting the quantum mechanical features of the processor by combining them with classical approaches.

2.2 Hybrid Solvers and the Constrained Quadratic Model (CQM)

As discussed above, even though any NP-complete problem can be mapped into an instance of the Ising model, the limitations of physical quantum annealing devices restrict the type of problems that can run on them. The sparse connectivity of superconducting quantum processors requires using multiple qubits to represent a single problem variable, reducing the problem instance size that can be solved. This becomes even more problematic for constrained problems. Since the native problem solved by quantum annealers is a Quadratic Unconstrained Binary Optimization (QUBO) problem, solving constrained problems first requires transforming the constraints into penalty terms that need to be added to the objective function. Mathematically, this can be done in a way that guarantees that the optimal solution of the resulting QUBO can be used to extract the optimal feasible solution of the original problem. However, this brings new issues: (i) enforcing equality constraints as penalty terms results in an increasing connectivity requirement (usually full connectivity); (ii) enforcing inequality constraints requires using slack variables, which need to be represented by physical qubits, further reducing the effective size of the instances that can be solved; (iii) guaranteeing the faithful mapping of optimal solutions can require fine-tuning of the penalty terms coefficients; and (iv) the required strength of the penalty terms may be beyond the dynamic range of the programmable parameters of the quantum annealer, resulting in some of the smaller coefficients in the objective function being effectively neglected. Most of these issues, which can have a negative effect on the performance of the device, also apply to other platforms like a digital annealer.

One way to get around these obstacles is to abandon the goal of a faithful mapping of optimal solutions and exploit the potential computational power of the quantum annealer in a hybrid framework.

The main idea of a hybrid solver is to pair the quantum processor with a classical heuristic in an iterative fashion, where this heuristic explores the search space and the output of quantum queries sent to the quantum module (QM) is used to suggest more promising areas of the solution space that the classical heuristic can further explore. A front-end takes the problem description as input and runs multiple threads of classical heuristics; the QM generates native QUBO problems that can be solved by the Quantum Processor Unit (QPU) using quantum annealing; the results are passed back to the front-end that outputs the best solution after a user-provided time limit. Figure 1 (from [14]) shows a schematic of the hybrid solver framework of D-Wave.

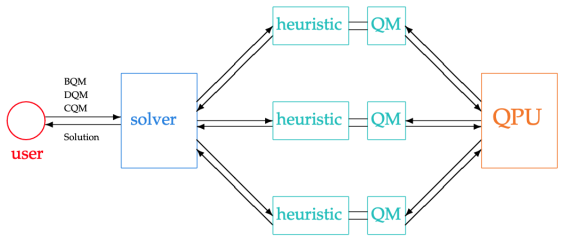

Figure 1: Schematic of the D-Wave hybrid solver framework.

D-Wave provides a Python framework to construct classical-quantum hybrid solvers. They also provide three user-ready hybrid solvers: the Binary Quadratic Model (BQM), the Discrete Quadratic Model (DQM), and the Constrained Quadratic Model (CQM). In this work, we used the CQM to solve the JO optimization problem. The main advantage of the CQM is its ease of use in programming complex constrained models. The inputs required are a description of the optimization variables (that could be binary or integer), the objective function, and the constraints. The constraints can be linear or quadratic, and the CQM provides simple tools to input any equality or inequality constraint. Algorithm 1 shows the steps required to solve a constrained quadratic problem. There are no steps required to map the problem to the underlying processor's connectivity, and there is no need to fine-tune parameters.

Algorithm 1 Constrained Quadratic Model

```
1: Declare the optimization variables
2: Describe the objective function to be minimised $f(x)$
3: Describe the constraints in terms of the declared variables
4: Initialize the CQM Model
5: Run the hybrid CQM solver
6: Extract the best solution provided that satisfies the constraints
```

2.3 Join Order (JO) Modeling and its QUBO Encoding

2.3.1 Join Order Modeling. We follow the NP-complete characterization provided in [28, 29] for the JO problem in terms of *problem input*, *solution space*, and *cost function*.

For the *problem input*, each query is represented in a graph form $Q = (V, E)$, where V is a set of nodes corresponding to the set of input relations (each node v_i represents a relation r_i) and E is a set of edges corresponding to the predicates used to join these relations (each edge e_{ij} represents a join predicate p_{ij} between two relations r_i and r_j). Each relation r_i has an n_i cardinality and each predicate p_{ij} is labeled with a respective join selectivity f_{ij}. In addition, there is no restriction on the query graph shape (e.g., cyclic, tree, star).

For the *solution space*, similar to existing works (e.g., [28, 29, 36]), we focus on the left-deep join tree solutions, where each solution has leaf nodes representing the input relations and intermediary nodes representing the join actions.

For the *cost function*, we assign a cost to a join between two relations r_i and r_j using the following classical cost function [6, 29]:

$$C_{\text{out}}(n_i, n_j) = f_{ij} n_i n_j \tag{2}$$

To assign a cost for the whole join tree, we apply C_{out} on the joins defined over the sequence s of relations $s_1, \ldots, s_n$ as follows:

$$C(s) := \sum_{i=2}^{n} C_{\text{out}}(|s_1 \ldots s_{i-1}|, |s_i|) \tag{3}$$

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

2.3.2 Join Order QUBO Encoding. State-of-the-art JO techniques, whether quantum-based (e.g., [25, 28]) or quantum-inspired (e.g., [29]), typically transform the problem into the QUBO form. Recently, in [29], the authors have proposed the most efficient QUBO-tailored encoding for the JO problem that (1) enforces valid join trees as solutions through constraints and (2) captures the cost of join operations more efficiently with the annealer. We adopt this encoding technique in our work. In particular, to have valid join trees, two main validity constraints are defined in [29]:

- *Constraint 1*: "Starting at two relations, the number of relations serving as operands for a join strictly increases by one with each additional join step"
- *Constraint 2*: "Once used as an operand for a join at step j, a relation must moreover serve as an operand for all joins directly or indirectly succeeding j"

Since these constraints were implemented in a QUBO form, they were encoded as quadratic penalty functions where a penalty weight A was applied if any constraint was not satisfied, inheriting the QUBO limitations mentioned in Section 2.2.

To encode the join order costs, [29] proposed a novel *quadratic cost approximation* method to exploit the QUBO's quadratic operations. First, it derives the *logarithmic intermediate result sizes* for the join tree. Then, it approximates their actual cardinalities as a *quadratic cost function*. Assume roj_{rj} is a binary variable defined for each pair of relation r and the join at step j and indicates whether a specific relation r is selected to participate in the join at step j. Similarly, assume paj_{pj} is a binary variable indicating whether a predicate p is applicable for the join at step j. A logarithmic intermediate result size LogIntCard(j) for a join j can be defined as follows:

$$\text{LogIntCard}(j) = \sum_{r=1}^{\text{R}} \text{LogCard}(r) \cdot \text{roj}_{rj} + \sum_{p=1}^{\text{P}} \text{LogPredSel}(p) \cdot \text{paj}_{pj} \tag{4}$$

where $\text{LogCard}(r)$ and $\text{LogPredSel}(p)$ are variable coefficients, providing the logarithmic cardinality for relation r and the logarithmic selectivity for predicate p, respectively, R is the number of relations, and P is the number of join predicates. Given the $\text{LogIntCard}(j)$ definition, a quadratic join cost function QCost can be defined as follows:

$$QCost = \sum_{t=1}^{T} \sum_{j=1}^{J} \theta_t \left(\text{Buffer}_{tj} - \text{LogIntCard}(j)\right)^2 \tag{5}$$

where Buffer_{tj} is given by the expression:

$$\text{Buffer}_{tj} = \sum_{i=1}^{N} (2^{i-1} \cdot \text{st}_{ij}) \tag{6}$$

Here, st_{ij} is a binary variable, and θ_t is one of T threshold values that will be contributing to the overall cost if $\text{LogIntCard}(j) \leq \log(\theta_t)$. The value of N is chosen such that Buffer_{tj} can take any value up to $\log(\theta_t)$. This setting guarantees that if $\text{LogIntCard}(j) \leq \log(\theta_t)$, there is a variable setting that minimizes the $QCost$ formula by assigning $\text{Buffer}_{tj} = \text{LogIntCard}(j)$, which eliminates the inner quadratic term in $QCost$. On the other hand, if $\text{LogIntCard}(j) > \log(\theta_t)$, Buffer_{tj} is unable to match $\text{LogIntCard}(j)$, leading to an increased cost.

3 MAIN APPROACH

3.1 CQM-based Constraints for Join Order

In our approach, we address the major issue in defining constraints for the JO optimization problem in QUBO encoding. In QUBO, the constraints for the optimization problem can only be defined as a quadratic penalty function added to the main cost function (check Section 2.2). Writing constraints in such a way that each violation results in a cost, thereby steering the solution towards feasibility. It is essential to select the appropriate penalty weights to ensure that these constraints are met, however, doing so without overwhelming the objective function can be difficult. If the penalty weights are set too high, they might dominate the objective function, forcing the solution to emphasize meeting constraints rather than optimizing the cost function. Conversely, if the weights are too low, the solver may disregard the constraints. Furthermore, digital and quantum annealers have low precision, making it impossible to distinguish between closely spaced energy levels, which may result in constraint breaches. Finally, as shown later, the JO optimization problem requires encoding inequality constraints. However, due to the limitation of not dealing with inequalities in typical quantum-annealing-based (e.g., [28]) and quantum-inspired (e.g., [29]) approaches, constraints are converted to equality function first, by using slack variables, and then included as penalty terms in the cost function. This is very expensive in terms of extra variables and can further hamper the quality of the solution.

To overcome the above-mentioned limitations, we propose using the CQM module provided by the D-Wave hybrid solver framework [14] to represent the different JO constraints. As mentioned in Section 2.2, the CQM provides the flexibility of defining constraints directly without adding any penalty function to the cost function or adding a penalty weight. The details of our proposed CQM-based constraints for the JO problem are below.

3.1.1 *CQM-based Constraint 1.*

Revisiting *Constraint 1* from Section 2.3.2, which enforces the shape of the left-deep join tree, we can easily provide a CQM equality constraint that checks, at join step j, if the total number of relations selected to participate in the join so far is exactly $j + 1$. This will ensure that each join step progressively involves more relations. The mathematical expression for this constraint is:

$$\sum_{r=1}^{\text{R}} \text{roj}_{r,j} = j + 1 \tag{7}$$

where R is the total number of relations, $j \in \{1, .., J\}$, and J is the total number of subsequent join steps in the query.

3.1.2 *CQM-based Constraint 2.*

Revisiting *Constraint 2* from Section 2.3.2, which enforces the continuity of relations selection throughout the sequence of join operations, we can provide a CQM in-equality constraint that checks, for each relation r, if this relation selected to participate in the join at step j is also selected to participate in the join at step $j + 1$. The mathematical expression for this constraint is:

$$\text{roj}_{r,j} - \text{roj}_{r,j+1} \leq 0 \tag{8}$$

Note that satisfying this constraint for all possible pairs of r and j ensures that if a relation r is selected at join step j, it will also appear in all subsequent join steps from $j + 1$ to J.

3.1.3 *CQM-based Constraint 3.*

A critical requirement to meet when setting rules for whether two relations should be joined or not is that a predicate applies to a join only if both relations involved in the predicate are members of that join. We can mathematically express this requirement through the following two constraints:

$$\text{paj}_{p_{(\text{r1},\text{r2})},j} - \text{roj}_{\text{r1},j} \leq 0 \tag{9}$$

$$\text{paj}_{p_{(\text{r1},\text{r2})},j} - \text{roj}_{\text{r2},j} \leq 0 \tag{10}$$

where $p_{(\text{r1},\text{r2})} \in \{1, .., P\}$ represents a join predicate defined over the two relations r1 and r2.

3.2 Overall CQM-based Join Order Algorithm

Algorithm 2 shows our end-to-end CQM-based algorithm for finding the optimal JO. The algorithm first initializes the CQM model, creates all required variables, and CQM-based constraints defined over them (lines from 7 to 20). Then, given these constraints, it optimizes the JO objective function using CQM (line 21). We have adopted the same quadratic cost approximation method from [29] as an objective function. The current CQM solver utilizes the LeapHybridCQMSampler framework [7] on D-Wave Quantum Annealers. LeapHybridCQMSampler is built to solve large-scale CQM problems and employs a hybrid quantum-classical approach. While quantum computers excel in swiftly exploring a huge range of potential solutions due to quantum parallelism, they are augmented by classical processors that handle higher-level issue decomposition and solution synthesis. This hybrid framework seeks the benefits of both technologies: the speed of quantum processing and the dependability and scalability of traditional algorithms.

Algorithm 2 Constrained Quadratic Model For Join Order Optimization

```
1: Input: Cardinalities, Selectivities, and Relations
2: Output: Final Join Order, Final Cost
3: procedure OPTIMIZE JOIN ORDER
4:     Initialize $cardinalities$ and $selectivities$ from Input
5:     $log\_cardinalities \longleftarrow \{\log(cardinalities)\}$
6:     $log\_selectivities \longleftarrow \{\log(selectivities)\}$
7:     Initialize CQM Model
8:     Define binary variables $paj\_vars$ and $roj\_vars$
9:     for each relation $r$ and join $j$ do
10:        Add constraint for incremental join participation
11:        CQM-based Constraint 1: $\sum_{r=1}^{R} roj_{r,j} = j + 1$
12:        Add constraint for continuity of relations
13:        CQM-based Constraint 2: $roj_{r,j} - roj_{r,j+1} \leq 0$
14:    end for
15:    for each predicate $p$ and join $j$ do
16:        Add constraints for predicate applicability for $r_1$
17:        CQM-based Constraint 3: $paj_{p_{(r1,r2)},j} - roj_{r1,j} \leq 0$
18:        Add constraints for predicate applicability for $r_2$
19:        CQM-based Constraint 3: $paj_{p_{(r1,r2)},j} - roj_{r2,j} \leq 0$
20:    end for
21:    Solve objective function (Equation 5) in CQM
22:    Extract $final\_join\_order$
23:    Calculate $final\_cost$ using Equation 3
24:    return $final\_join\_order, final\_cost$
25: end procedure
```

4 EXPERIMENTAL EVALUATION

4.1 Experimental Setup

Our objective is to show the efficiency of our approach, referred to as CQM, in finding more optimal JO sequences compared to the original QUBO encoding in [29], referred to as DA method in this section, given the same optimization time window. Note that although the results reported by DA in [29] were based on the Fujitsu digital annealer (a quantum-inspired hardware accelerator for computational tasks), they dominated the results provided by other quantum-based competitors in most cases. Therefore, we compare our CQM results to the DA ones[1]. To have a fair comparison, we set the time for the solver execution to be a maximum of 60 seconds, as proposed in the experimental setup of DA in [29], to provide the model enough time to find an optimal solution. We also stick to the same evaluation method in [29]: (1) calculate the join cost (Equation 3) of the obtained JO sequences based on the intermediate result sizes, and then (2) normalize the costs (relative to the best solution; square root scale).

Our CQM optimization algorithm is implemented in Python 3.12 and constructed using the D-Wave Ocean toolbox 'dimod' library (version 0.12.14). We run the algorithm using the D-Wave's Leap Hybrid CQM Sampler [7]. We used all the default configurations by D-Wave except the time of the execution solver, which we set as mentioned earlier. For cost threshold values θ_t in Equation 5, we used the same values reported in [29].

[1]The results reported by the DA authors exist in the following open-source repository: https://github.com/lfd/vldb24/tree/main

Regarding benchmarks, we ran our experiments with three workloads provided by the authors of the paper [29][2]:

- 'JOB': The Join Order Benchmark (JOB) contains queries on the IMDB dataset [18].
- 'Chain' and 'Cycle': Two synthetic join benchmarks, proposed in [29], including join queries of chain and cycle graph structures with different numbers of relations. These benchmarks were designed to simulate real-world usage scenarios.

4.2 Experimental Results

Generally speaking, CQM is able to produce an overall improvement in the JO cost efficiency for the three workloads (each workload query has up to 50 relations to join) while finding the optimal JO solution over the same period of optimization time used in [29].

Figures 2, 3, and 4 show our comparison results in the JOB[3], Chain, and Cycle workloads, respectively. We observe that CQM outperforms DA in almost all queries. There are few instances in which the results difference is of decimal points due to which the minimum improved efficiency is presented as 0%. As we can see, for JOB Workload, CQM method significantly outperforms DA in many queries. In the case of Chain and Cycle workloads, the results show less improvement in comparison to what we get in the case of JOB workload, yet there are still improvements over DA. These results indicate that CQM method is more effective when the join queries are more complex, as in the JOB workload.

Table 1 has also presented a summary of our results. For the JOB workload, the minimum cost efficiency gained is 1%, and the highest recorded is up to 98%. On the other hand, in the case of Chain and Cycle workloads, the final cost results for some queries had very little difference, so the overall cost efficiency of the CQM method over DA is calculated as 0%.

In brief, our results show that running the JO problem with a hybrid quantum-classical approach is beneficial and provides room for exploring different ways to implement the problem, potentially yielding more significant results. So far, we have utilized the quadratic cost approximation function provided in [29]. However, our results have prompted us to further pursue analytical research to develop approximation models for the JO cost that would potentially be suitable for executing on hybrid classical-quantum solvers.

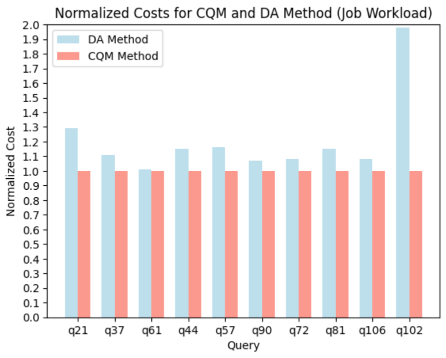

Figure 2: Normalized join costs (relative to CQM) of both CQM and DA methods for JOB workload.

[2]We used the open-source implementations of these workloads provided in https://github.com/lfd/vldb24/tree/main

[3]For the job workload, we present only the results for 10 queries to avoid cluttering.

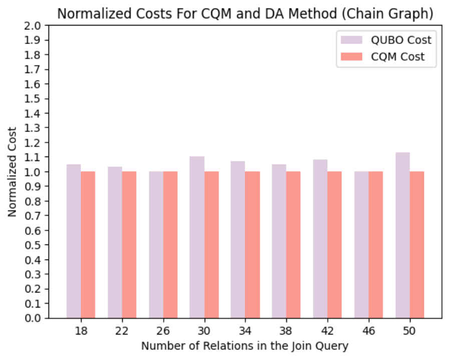

Figure 3: Normalized join costs (relative to CQM) of both CQM and DA methods for Chain workload.

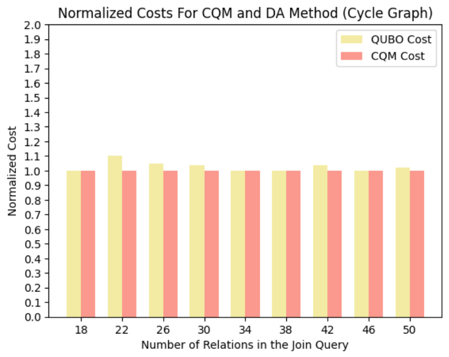

Figure 4: Normalized join costs (relative to CQM) of both CQM and DA methods for Cycle workload.

Table 1: Measures of Percentage Cost Improvement For Three Different Query Loads

Statistic	Job Workload	Chain Graph	Cycle Graph
Minimum	1.0%	0.0%	0.0%
Maximum	98.0%	13.0%	10.0%
Average	20.8%	5.7%	2.8%
Median	13.0%	5.0%	2.0%

5 RELATED WORK

The join order (JO) problem is one of the most thoroughly researched problems in query optimization, as evidenced by numerous studies (e.g., [19, 26, 27, 36]). Many approaches have been proposed to exploit dynamic programming (e.g., [24, 31, 37]) to find the optimal JO solution. However, such approaches struggle with scalability as the number of relations increases. On the other hand, heuristic-based approaches (e.g., [4, 13, 15, 32, 33]) provide more scalable solutions when handling large join queries, yet still bounded by the quality-efficiency tradeoff when running on classical computers. More recent efforts exploit machine learning to learn better JO (e.g., [5, 22, 23]) or better query optimization plans in general (e.g., [9, 21, 22, 39]).

Many quantum-based techniques (e.g., [25, 28, 38]) have been proposed to address the JO problem. [28] investigated the performance of the QUBO formulation of the JO problem provided in [36] using gate-based quantum computing and quantum annealing. However, such QUBO formulation focused on the left-deep join tree cases only. A more generic QUBO encoding has been proposed in [25] to handle bushy join trees among all possible JO solutions. Recently, a more efficient QUBO encoding for the JO problem, exploiting both linear and quadratic terms in the QUBO representation, has been introduced for left-deep join trees in [29], and extended for the general join trees in [30]. We decided to extend the encoding in [29] using the CQM constraints instead of [30] because [29] had more thorough benchmarking results that we can compare the performance of our CQM-based approach against ([30] is just a theoretical approach, and we plan to extend it with our approach in the future). Another interesting direction [38] has explored using quantum machine learning for JO optimization as well. Apart from JO, QUBO encoding has been investigated in other database problems such as multi-query optimization [35] and transaction scheduling [3, 12].

The direction of investigating hybrid classical-quantum approaches has been paid very little attention in our database community. To the best of our knowledge, [10] is the only work exploring such direction in the multi-query optimization problem. However, there is no work investigating the power of CQM-based hybrid solvers, like the one provided by D-Wave [14]. We are the first to fill this gap, where we used the JO problem as a case study to show the effectiveness of these hybrid solvers.

6 CONCLUSION AND FUTURE WORK

Our preliminary results show that a very basic implementation of a classical-quantum hybrid solver like CQM, can produce performance results that already improve those of other methods reported in the literature, like the ones obtained using a quantum-inspired digital annealer. The average improvement on the cost of the best join order found by CQM ranges from $\sim 3\%$ to about $\sim 6\%$ for two synthetic benchmarks ('Chain' and 'Cycle'), while reaching up to 20% for the JOB benchmark, that contains real-world queries.

It is worth noting that the CQM is a very user-friendly solver that does not require the user to take care of the complicated mapping of the problems into the quantum processor. An instance with many linear (and even quadratic) constraints can be input into the solver very straightforwardly. The solver then applies a set of classical heuristics combined with quantum queries to the quantum processing unit (QPU) to accelerate the exploration of the space of feasible solutions. That being said, one drawback is that pre-processing the problem instances and generating the native QPU problems by CQM is still a black box, and hence, fully optimizing the JO encoding process is not entirely fulfilled. Nevertheless, the CQM-based results obtained in this paper are competitive with other methods. This clearly calls for continued research on constructing smarter hybrid classical-quantum solvers that could further improve the performance. Our future plan is to exploit the D-Wave's framework [7] for implementing user-designed hybrid solvers in order to design a JO-specific solver that captures the features of the JO problem and embeds them in the hybrid classical-quantum operations. We also plan to have a detailed analysis for the JO optimization running time of CQM, extend our JO investigation to explore bushy and general join trees (similar to [25, 30]), and even go beyond JO and explore CQM with other database operations.

REFERENCES

[1] F Barahona. 1982. On the computational complexity of Ising spin glass models. *Journal of Physics A: Mathematical and General* 15, 10 (oct 1982), 3241. https://doi.org/10.1088/0305-4470/15/10/028

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

[3] Tim Bittner and Sven Groppe. 2020. Avoiding Blocking by Scheduling Transactions using Quantum Annealing. In *IDEAS*.

[4] 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

[5] Tianyi Chen, Jun Gao, Hedui Chen, and Yaofeng Tu. 2023. LOGER: A Learned Optimizer Towards Generating Efficient and Robust Query Execution Plans. *Proc. VLDB Endow.* 16, 7 (mar 2023), 1777–1789. https://doi.org/10.14778/3587136.3587150

[6] Sophie Cluet and Guido Moerkotte. 1995. On the complexity of generating optimal left-deep processing trees with cross products. In *Database Theory — ICDT '95*, Georg Gottlob and Moshe Y. Vardi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 54–67.

[7] D-Wave Leap's Hybrid Solvers. 2024. https://docs.dwavesys.com/docs/latest/doc_leap_hybrid.html.

[8] D-Wave documentation. 2024. https://docs.dwavesys.com/docs/latest/c_gs_4.html.

[9] L. Doshi et al. 2023. Kepler: Robust Learning for Parametric Query Optimization. In *SIGMOD*.

[10] Tobias Fankhauser, Marc E. Solèr, Rudolf M. Füchslin, and Kurt Stockinger. 2021. Multiple Query Optimization using a Hybrid Approach of Classical and Quantum Computing. *CoRR* abs/2107.10508 (2021). arXiv:2107.10508 https://arxiv.org/abs/2107.10508

[11] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 [quant-ph]

[12] Sven Groppe and Jinghua Groppe. 2021. Optimizing Transaction Schedules on Universal Quantum Computers via Code Generation for Grover's Search Algorithm. In *Proceedings of the 25th International Database Engineering & Applications Symposium* (Montreal, QC, Canada) *(IDEAS '21)*. Association for Computing Machinery, New York, NY, USA, 149–156. https://doi.org/10.1145/3472163.3472164

[13] 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

[14] Hybrid Solver for Constrained Quadratic Models [WhitePaper]. 2021. https://www.dwavesys.com/media/rldh2ghw/14-1055a-a_hybrid_solver_for_constrained_quadratic_models.pdf.

[15] Y. E. Ioannidis and Younkyung Kang. 1990. Randomized algorithms for optimizing large join queries. *SIGMOD Rec.* 19, 2 (may 1990), 312–321. https://doi.org/10.1145/93605.98740

[16] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. 2011. Quantum annealing with manufactured spins. *Nature* 473, 7346 (2011), 194–198. https://doi.org/10.1038/nature10012

[17] Tadashi Kadowaki and Hidetoshi Nishimori. 1998. Quantum annealing in the transverse Ising model. *Phys. Rev. E* 58 (Nov 1998), 5355–5363. Issue 5. https://doi.org/10.1103/PhysRevE.58.5355

[18] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proc. VLDB Endow.* 9, 3 (nov 2015), 204–215. https://doi.org/10.14778/2850583.2850594

[19] 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, 5 (oct 2018), 643–668. https://doi.org/10.1007/s00778-017-0480-7

[20] Andrew Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2 (2014). https://doi.org/10.3389/fphy.2014.00005

[21] R. Marcus et al. 2021. Bao: Making Learned Query Optimization Practical. In *SIGMOD*.

[22] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: A Learned Query Optimizer. *Proc. VLDB Endow.* 12, 11 (jul 2019), 1705–1718. https://doi.org/10.14778/3342263.3342644

[23] Ryan Marcus and Olga Papaemmanouil. 2018. Deep Reinforcement Learning for Join Order Enumeration. In *Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management* (Houston, TX, USA) *(aiDM'18)*. Association for Computing Machinery, New York, NY, USA, Article 3, 4 pages. https://doi.org/10.1145/3211954.3211957

[24] 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

[25] Nitin Nayak, Jan Rehfeld, Tobias Winker, Benjamin Warnke, Umut Çalikyilmaz, and Sven Groppe. 2023. Constructing Optimal Bushy Join Trees by Solving QUBO Problems on Quantum Hardware and Simulators. In *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments (BiDEDE '23)*. Association for Computing Machinery, New York, NY, USA, Article 7, 7 pages. https://doi.org/10.1145/3579142.3594298

[26] 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

[27] 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

[28] Manuel Schönberger, Stefanie Scherzinger, and Wolfgang Mauerer. 2023. Ready to Leap (by Co-Design)? Join Order Optimisation on Quantum Hardware. *Proc. ACM Manag. Data* 1, 1, Article 92 (may 2023), 27 pages. https://doi.org/10.1145/3588946

[29] Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. 2023. Quantum-Inspired Digital Annealing for Join Ordering. In *VLDB*.

[30] Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. 2023. Quantum Optimisation of General Join Trees. In *Joint Proceedings of Workshops at the 49th International Conference on Very Large Data Bases (VLDB 2023), Vancouver, Canada, August 28 - September 1, 2023 (CEUR Workshop Proceedings, Vol. 3462)*. CEUR-WS.org. https://ceur-ws.org/Vol-3462/QDSM2.pdf

[31] 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

[32] 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

[33] Arun Swami and Anoop Gupta. 1988. Optimization of large join queries. *SIGMOD Rec.* 17, 3 (jun 1988), 8–17. https://doi.org/10.1145/971701.50203

[34] The D-Wave Advantage System [WhitePaper]. 2022. https://www.dwavesys.com/media/3xvdipcn/14-1058a-a_advantage_processor_overview.pdf.

[35] Immanuel Trummer and Christoph Koch. 2016. Multiple Query Optimization on the D-Wave 2X Adiabatic Quantum Computer. In *VLDB*.

[36] 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* (Chicago, Illinois, USA) *(SIGMOD '17)*. Association for Computing Machinery, New York, NY, USA, 1025–1040. https://doi.org/10.1145/3035918.3064039

[37] 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

[38] Tobias Winker, Umut Çalikyilmaz, Le Gruenwald, and Sven Groppe. 2023. Quantum Machine Learning for Join Order Optimization using Variational Quantum Circuits. In *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments* (<conf-loc>, <city>Seattle</city>, <state>WA</state>, <country>USA</country>, </conf-loc>) *(BiDEDE '23)*. Association for Computing Machinery, New York, NY, USA, Article 5, 7 pages. https://doi.org/10.1145/3579142.3594299

[39] Z. Yang et al. 2022. Balsa: Learning a Query Optimizer Without Expert Demonstrations. In *SIGMOD*.