Index Advisors on Quantum Platforms

Manish Kesarwani
IBM Research & Indian Institute of Science
Bangalore, India
manishkesarwani@in.ibm.com

Jayant R. Haritsa
Indian Institute of Science
Bangalore, India
haritsa@iisc.ac.in

ABSTRACT

Index Advisor tools settle for sub-optimal index configurations based on greedy heuristics, owing to the computational hardness of index selection. We investigate here how this limitation can be addressed by leveraging the computing power offered by quantum platforms. Specifically, we present a hybrid Quantum-Classical Index Advisor that judiciously incorporates gate-based quantum computing within a classical index selection wrapper.

Two distinct trade-offs between solution quality and computational complexity are considered. First, index selection is modeled as a Quadratic Unconstrained Binary Optimization problem and solved using the popular Quantum Approximate Optimization Algorithm. The obtained solution is approximate, like greedy, but significantly better in quality while incurring only $O(\log(L))$ computations, where L is the total number of candidate configurations. Second, index selection is modeled as a fully enumerative search and solved using the seminal Grover Search algorithm. A novel quantum oracle is proposed that performs computations on data hosted in the relative phase of a quantum superposition state, and is encoded using only standard quantum gates. This approach identifies, with high probability, the *optimal* index configuration with $O(\sqrt{L})$ computations.

We have implemented these two designs using the Qiskit SDK and performed proof-of-concept evaluations on both simulation and hardware platforms. Substantive quality improvements, by a multiplicative factor of 1.5 to 2 and approaching optimality, are obtained as compared to a commercial database engine implementing a greedy approach. Moreover, their quantum resource requirements effectively scale linearly with problem size, an essential feature from a feasibility perspective.

PVLDB Reference Format:
Manish Kesarwani and Jayant R. Haritsa. Index Advisors on Quantum Platforms. PVLDB, 17(11): 3615 - 3628, 2024.
doi:10.14778/3681954.3682025

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

Proceedings of the VLDB Endowment, Vol. 17, No. 11 ISSN 2150-8097.
doi:10.14778/3681954.3682025

1 INTRODUCTION

Given an SQL query workload, the creation of appropriate column indexes has been a standard database technique for materially reducing the workload's execution time. In the early days, these indexes were manually selected by DBAs. However, contemporary engines feature automated Index Advisors (IA) that identify good configurations while adhering to storage budgets; for instance, IBM's DB2 Index Advisor [52] and Microsoft's AutoAdmin [19].

Given their demonstrated performance impact, it is no surprise that IA design has been an active area of research over the past three decades in both academia and industry (e.g. [7, 12, 13, 16–19, 21, 22, 32, 47, 48, 52, 58]), with even machine learning-based techniques [8, 36, 38, 44] appearing in recent times. However, searching for an optimal index configuration inherently entails exploring a combinatorially large search space. Therefore, current advisors typically rely on heuristic strategies, which can result in suboptimal index configurations. For instance, DB2 IA reduces the index selection problem to an instance of a 0-1 Knapsack Problem [46], and then invokes a greedy heuristic-based solver to recommend the index configuration. The heuristic is essentially "ROI" (return on investment) – the time benefit provided by the index normalized to its storage footprint.

Problem Instance

Index (I)	Columns in the Index	Time Benefit (V)	Storage Cost (W)
i_0	[L_DISCOUNT, L_SHIPDATE, L_QUANTITY, L_EXTENDEDPRICE]	165811	266
i_1	[L_SHIPDATE, L_DISCOUNT, L_EXTENDEDPRICE, L_PARTKEY]	178871	232
i_2	[P_CONTAINER, P_BRAND, P_PARTKEY]	1213770	8
i_3	[L_PARTKEY, L_QUANTITY]	1213770	132
i_4	[L_PARTKEY, L_QUANTITY, L_EXTENDEDPRICE]	1213770	199
i_5	[C_ACCTBAL, C_CUSTKEY, C_PHONE]	44370	2
i_6	[O_CUSTKEY]	44370	9

***Storage Capacity*: 140**

CDB Recommendation	***Optimal Configuration***
Heuristic Index Set: $\{i_2, i_5, i_6\}$	*Optimal Index Set:* $\{i_2, i_3\}$
Heuristic Benefit: 1302510	*Optimal Benefit: 2427540*

Figure 1: Suboptimality of Heuristic IA

Consider an SQL workload comprising TPC-H [50] queries Q6, Q14, Q22, and two instances of Q17 over a 1GB TPC-H database. Given this setup, the index selection problem instance generated by a popular commercial database (CDB) engine is shown in Figure 1. The problem instance comprises seven candidate indexes, their expected time benefit wrt query response time, and storage cost overhead. Now, for a storage budget of 140, CDB recommends a sub-optimal index configuration $\{i_2, i_5, i_6\}$ with a total benefit of 1302510, while the optimal configuration comprises indexes $\{i_2, i_3\}$ with a benefit of 2427540. In this scenario, it is evident that around 50% of the available index benefit is lost due to a sub-optimal choice.

As highlighted in [12], a variety of heuristics have been proposed to bridge this gap to optimality – however, there is no guaranteed improvement. Therefore, we take a radically different approach here: specifically, we ask "Is it feasible to utilize the raw computational power promised by *quantum platforms* to find better, perhaps even optimal, solutions?". And the good news, as explained in the remainder of this paper, is that it indeed appears viable, through careful algorithmic design, to concurrently achieve excellent quality and practical efficiency. As a case in point, the optimal configuration for the ~ 50 candidate indexes constructable on a TPC-H database can be determined, with high probability, in a few hours on a quantum computer (as per our estimated projection in Section 6.5.2). In contrast, an exhaustive search would take a couple of *months* on current hardware.

Our study is motivated by the growing interest in early-stage quantum computing with 100,000-qubit machines projected within the coming decade [28]. Even within the database community, quantum computing has begun to attract attention – similar, albeit unrelated, studies to ours have recently been carried out for join-order optimization [42, 49, 55] and transaction scheduling [9, 10, 29], with promising outcomes. Finally, a call for quantum implementation of Index Advisors was explicitly advocated in recent database vision papers [31, 56].

Quantum Modeling Dimensions

A variety of design choices and challenges arise in porting IA to the quantum domain. First, there are alternative computing models – *quantum annealing*, which is energy-minimization-based, and *quantum circuit*, which is gate-based, similar to classical circuits. Over the past decade, the latter has gained prominence as it provides a greater degree of design flexibility [49], and we therefore focus on this choice in our work.

Second, we could ask whether the IA design should be purely quantum or a hybrid that synergistically leverages classical and quantum computing. We have chosen the latter to (a) facilitate easy integration with contemporary DBMS engines and (b) minimize the quantum circuit complexity to address only the hard computational problems.

Third, quantum computers work in *probabilistic* space – this means that individual results may have errors or even violate mandatory constraints. We therefore need to devise schemes to eliminate, with at least high confidence, these inherent problems of quantum computation.

Fourth, whether IA should be modeled as a *optimization* problem or as a *search* formulation. Due to the disparate trade-offs between their solution quality and computational effort, we design and evaluate both options. For the former, index selection is modeled as a Quadratic Unconstrained Binary Optimization (QUBO) problem, and solved using the Quantum Approximate Optimization Algorithm (QAOA) [24]. This approach requires $O(\log(L))$ computations, where L is the total number of candidate configurations, and the recommended configurations are significantly better compared to classical heuristics. On the other hand, for the latter, index selection is modeled as a fully enumerative search over the exponential configuration space, and solved using the seminal Grover Search algorithm [30]. This approach identifies, with high confidence, the *optimal* index configuration incurring $O(\sqrt{L})$ computations.

Finally, modeling database applications on quantum platforms poses tricky implementation issues. For instance, with Grover Search, we have to devise an efficient data loading scheme that can scale with the size of the problem and also facilitate subsequent computations. Our approach diverges from the conventional method of loading data via *basis states* – instead, we load the data in the *phase* of the qubits. This unconventional strategy allows for more efficient computation and paves the way for further innovations, such as intrinsic computation of aggregate weights and benefits of each index configuration, and easy identification of qualifying configurations by just checking the sign qubit. We also design an efficient quantum *oracle* that is able to identify qualifying configurations. The creation of such an oracle is a complex task, as it requires performing computations while the data is phase-resident in a quantum superposition state.

The QIA System

The overall architecture of our hybrid Quantum-Classical Index Advisor (QIA) framework is shown in Figure 2. Given a database environment, the system takes the SQL workload and storage budget as input and outputs a recommended index configuration.

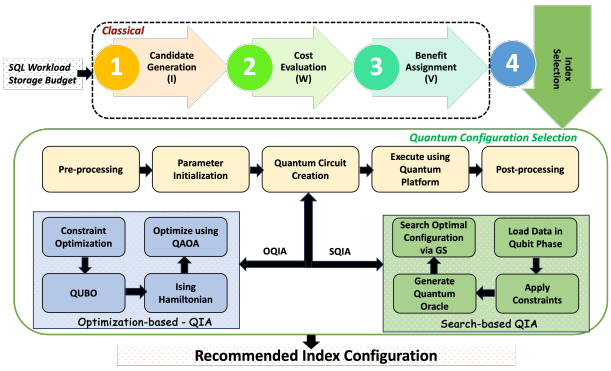


Figure 2: Quantum Index Advisor (QIA) Architecture

The first three steps – candidate configuration generation, followed by computing the storage costs and time benefits of these configurations – are carried out in the classical world, whereas the final computationally intensive index selection step is hosted on the quantum platform. Based on the user choice, either OQIA (Optimization-based QIA) or SQIA (Search-based QIA) is invoked to recommend the index configuration. With SQIA, the user provides an additional parameter, δ, the desired probability of obtaining the optimal solution. This parameter sets an exponential trade-off between result quality and search time, as quantified in Equation 10. While the OQIA porting is an amalgamation of known techniques, SQIA represents, to our knowledge, the first application of quantum search to the index selection problem implementable through standard quantum gates.

Evaluation

We have designed quantum circuits for the OQIA and SQIA algorithms and implemented them using the Qiskit SDK [33]. They have been evaluated on a **32-qubit** noiseless quantum simulator, and on a **127-qubit** IBM Eagle circuit processor. Given the current limited capacity of quantum platforms, we are perforce only able to model modest instances of the IA problem, which we have evaluated on the TPC-H environment.

Interestingly, however, we show that even for these modest instances, QIA obtains substantive quality improvements, approaching optimality. As an exemplar, consider again the problem instance of Figure 1, where greedy delivered 0.54 of the optimal (as determined by exhaustive search). For this scenario, OQIA produced a 0.76 solution, while SQIA recommended the optimal.

Moreover, these good results are obtained while incurring substantially less computational effort than exhaustive search. Specifically, with OQIA, the computational overheads are 23% relative to exhaustive search, while SQIA is 73%. Finally, we make the case that for large problem instances, our techniques effectively scale the qubit requirements *linearly* with problem size, an essential feature from a long-term feasibility perspective.

Index Advisor Framework

To put the Index Advisor framework into perspective, Figure 3 shows a high-level characterization of the QIA techniques, contrasted to the greedy and exhaustive approaches for challenging IA instances (like the one shown in Figure 1). The dimensions are the (normalized) index configuration quality, the computational efficiency in identifying these configurations, and the probabilistic distribution of the quality. On the efficiency axis, $L = 2^I$, where I is the number of indexes.

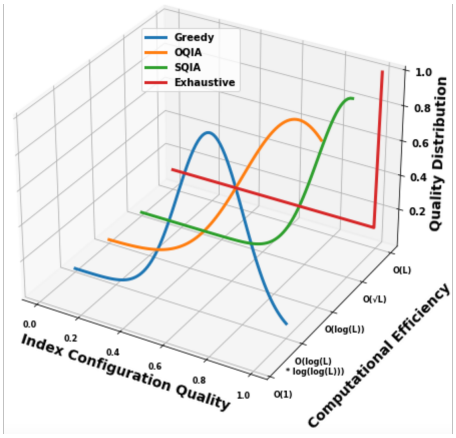


Figure 3: Quality-Efficiency Characterization

We see here that OQIA consistently delivers high-quality index configurations while performing fewer computations than Greedy. Further, by making a modest additional computational investment, SQIA could be instead used to obtain optimal configurations with a high probability. But we hasten to add that the computations performed by Greedy are simpler, and hence it may be empirically faster for contemporary index sizes. Therefore, a variety of quality-efficiency trade-offs are available, and in a complete deployment, a sentinel module would be required to make the appropriate choice. We plan to explore this aspect in our future work.

Contributions

To summarize, our contributions are the following:

(1) Proposed the first hybrid IA architecture that harnesses classical computing and gate-based quantum technology in a pluggable manner for database engines.
(2) Logical design and circuit representations for the OQIA and SQIA approaches. The SQIA design, in particular, represents an original application of quantum search to the index selection problem, leveraging a phase-resident approach to data storage while only employing standard quantum gates. Further, OQIA and SQIA provide different tradeoffs between configuration quality and computational efficiency.
(3) A pilot implementation and evaluation of the proposed IA approaches on both (noiseless) quantum simulators and (noisy) quantum circuit processors. The evaluations show that substantively improved index configurations, by a multiplicative factor of 1.5 to 2 and approaching optimality, are achievable through quantum technology. We also observe that OQIA is more robust to noise than SQIA.

To our knowledge, this study represents the first investigation of quantum computing to the IA problem. For this initial analysis, we restrict our attention to the computationally expensive index selection step in the IA pipeline. We intend to explore quantum implementation of other pipeline components in our future work.

The rest of the paper is organized as follows: A brief background of quantum data processing is given in Section 2. The formal problem framework is detailed in Section 3. Then, in Sections 4 and 5, we present the OQIA and SQIA algorithms, respectively, and their performance evaluation is profiled in Section 6. Related work is reviewed in Section 7. Finally, our conclusions and future research avenues are highlighted in Section 8.

2 QUANTUM BACKGROUND

Quantum computation brings to bear on information processing, the fundamental phenomena of quantum mechanics [43], such as *superposition*, *interference*, *entanglement*, *reversible computation*, and *irreversible measurements*. Here, we briefly review the basic building blocks used in QIA.

Quantum computation is built upon the *quantum bit* or *qubit*. The possible states for a qubit are $|0\rangle$ and $|1\rangle$ (in Dirac notation), which correspond to the states 0 and 1 of a classical bit. But unlike a classical bit, a qubit can be in a state $|\psi\rangle$ which is a linear combination, or superposition, of the $|0\rangle$ and $|1\rangle$ states:

$$|\psi\rangle = \alpha|0\rangle + e^{i\gamma}\beta|1\rangle \quad (1)$$

where α and β are complex numbers such that $|\alpha|^2+|\beta|^2 = 1$ and $\gamma \in [0, 2\pi)$ is the quantum phase (angle of rotation around the Z-axis). When a qubit is measured, we get either 0 with probability $|\alpha|^2$, or 1 with probability $|\beta|^2$, while the phase γ is lost. Furthermore, the measurement operation is irreversible, since it destroys the quantum superposition state and outputs classical bits.

2.1 Quantum Gates

A quantum algorithm generates a quantum circuit comprising elementary quantum gates wired together to accomplish a task. The quantum gates used in QIA are summarized below:

NOT Gate (X): Operates on a single qubit and is quantum equivalent of the classical NOT gate. It takes a state $\alpha|0\rangle+\beta|1\rangle$ as input and flips the amplitudes of $|0\rangle$ and $|1\rangle$, producing the state $\beta|0\rangle+\alpha|1\rangle$.

Hadamard Gate (H): Operates on a single qubit and produces an *equal* superposition of the $|0\rangle$ and $|1\rangle$ states. That is, it turns $|0\rangle$ into $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, and $|1\rangle$ into $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$.

Phase Gate (P): Takes an angle ϕ as input and rotates a qubit about the Z-axis, mapping: $|0\rangle \rightarrow |0\rangle$ and $|1\rangle \rightarrow e^{i\phi}|1\rangle$.

Controlled-Phase Gate (CP): Takes an angle ϕ as input and operates on a pair of qubits: a *control* qubit and a *target* qubit. The **P** gate is applied to the target qubit conditional on the state of the control qubit.

Multi-Controlled Toffoli Gate (MCT): Operates on a set of n qubits, with $n-1$ *control* qubits, and the remaining qubit being the *target*. If all control qubits are set to $|1\rangle$, then the target qubit is flipped; otherwise, it is left undisturbed. For $n = 2$, MCT reduces to the fundamental Controlled-NOT Gate (**CX**).

2.2 Quantum Approximate Optimization Algorithm (QAOA)

QAOA is a hybrid algorithm that combines classical and quantum components and is tailored to find approximate solutions to optimization problems [24]. QAOA operates through a sequence of quantum and classical steps. The quantum part involves initializing qubits in the uniform superposition state $|+\rangle$, and then applying a specialized quantum circuit (configured with $2p$ parameters) on the initial state p times. A quantum computer is used to evaluate the objective function, while a classical optimizer is used to update the $2p$ parameters. This iterative process is repeated until the classical optimizer converges. The protocol is shown in Figure 4.

As might be expected, the approximation quality of QAOA improves with p, but the circuit depth also grows linearly with p. Therefore, p is usually set to a small value to balance solution quality and quantum feasibility. In fact, even at the lowest circuit depth ($p = 1$), QAOA has non-trivial provable performance guarantees [25], and hence $p = 1$ has been used as a common assignment in the literature [49]. However, the literature also suggests that a logarithmic depth is anticipated to surpass classical optimizers [53]. Therefore, in our evaluation, we vary p in the range 1 to $\lceil \log(|I|) \rceil$, where I is the list of candidate indexes.

2.3 Grover Search (GS)

GS is a quantum algorithm designed to solve the unstructured search problem with high probability (WHP) [30]. Specifically, given an unordered list of N items, GS identifies a desired item WHP, using $O(\sqrt{N})$ iterations, as compared to the $O(N)$ probes incurred by the classical algorithms. During each iteration, GS leverages quantum properties to simultaneously check all N items, resulting in a quadratic speed-up compared to classical methods.

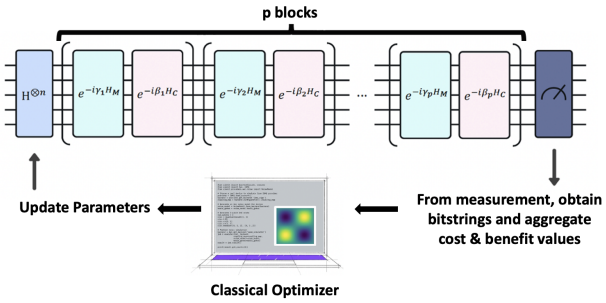


Figure 4: The QAOA Protocol [5]

The GS algorithm involves three key operators: *state preparation*, *quantum oracle*, and *diffusion*. The state preparation operator creates a quantum superposition state encompassing an exponential solution space. The quantum oracle enhances each candidate solution with problem-specific data, evaluates the cost function, and identifies qualifying candidates. The diffusion operator amplifies the measurement probability of the qualified candidates. Notably, the state preparation is a one-time activity. The combined application of the quantum oracle followed by the diffusion operator is termed a **Grover Iteration**. It is invoked multiple times to identify a qualifying candidate with a probability exceeding 0.5.

The key challenges in designing an index selection scheme based on GS include: (1) Creating an efficient problem-specific quantum oracle; (2) Identifying the precise number of Grover Iterations required for the GS algorithm to work appropriately; and (3) Boosting the success probability from 0.5 to the user-desired δ. In Section 5, we present a novel approach to construct the quantum oracle, utilizing various quantum concepts implementable through standard quantum gates. Additionally, we address the remaining challenges by adapting the Generalized Grover Search (GGS) [11, 14] algorithm and the Powering Lemma [34], which enable performance optimization. These modules are discussed in detail in Section 5.2.

2.4 Shots

An operational parameter that influences the solution quality of quantum algorithms is *"shots"(S)*. Shots indicate the number of times a quantum algorithm is executed, with increased shots providing more accurate and reliable results at the expense of consuming more quantum resources. In our experiments, we empirically identify the ideal number of shots for the proposed QIA schemes.

3 PROBLEM FRAMEWORK

The Index Advisor problem that we consider here is the following:

Given an SQL query workload $\mathcal{Q}$ on a relational database instance $\mathcal{D}$, recommend a configuration of indexes that maximizes the performance benefit for the workload while adhering to the following constraints: (1) **Space Constraint:** The configuration must fit in a user-specified storage budget; and (2) **Validity Constraint:** The configuration must satisfy validity requirements, which could be (a) intrinsic – for instance, at most one *clustered* index per relation, or (b) extrinsic – for instance, mandatorily add all indexes listed in a pre-specified base configuration.

In the above problem definition, the performance benefit of a configuration is measured as the reduction in the estimated execution time of the workload compared to the base configuration. Further, since contemporary database systems typically build, by default, a clustering index on the primary key column of each relation, we assume that the base configuration comprises these indexes. Therefore, our objective is restricted to selecting the additional *unclustered* indexes – however, selection of clustered indexes can also be incorporated in our framework, as detailed in [35].

Index Advisor Pipeline. The index advisor pipeline encompasses a sequence of tasks to find a beneficial configuration of constraint-compliant indexes. The tasks, shown pictorially in the top pipeline of Figure 2 as Steps 1 through 4, are the following:

(1) **Candidate Generation:** This task entails identifying candidate indexes to improve the SQL workload performance. Techniques such as analyzing query predicates and recognizing common query patterns, are employed to create a comprehensive pool of potentially beneficial indexes.
(2) **Cost Evaluation:** This step computes the individual storage and maintenance (due to updates) overheads of the candidate indexes. Storage overheads serve to model the index cost, whereas maintenance overheads are factored into the benefit calculations of the next stage.
(3) **Benefit Computation:** The overall improvement of the query workload execution time due to the presence of each index is computed, typically via the query optimizer module. Specifically, the cumulative improvement in query response times is weighed against the increase in index maintenance overheads.
(4) **Index Selection:** This final step aims to find the configuration among the candidate indexes that maximizes the benefit while respecting the storage and validity constraints.

Our study employs classical strategies for the first three tasks in the pipeline and uses the quantum platform only for the final computationally-intensive index selection task. Specifically, we use DB2 Index Advisor [52] as the exemplar classical technique.

In this formulation, given a set of indexes $I = \{i_0, i_1, \cdots, i_{n-1}\}$, each with its storage overhead $W = \{w_0, w_1, \cdots, w_{n-1}\}$ and time benefit $V = \{v_0, v_1, \cdots, v_{n-1}\}$, index selection in Step 4 is modeled as the following constrained optimization problem:

$$\max \sum_{i=0}^{n-1} x_i v_i \quad \text{s.t.} \quad \sum_{i=0}^{n-1} x_i w_i \leq W_{max} \tag{2}$$

where the solution to the problem is represented by an array, denoted as X, consisting of elements $\{x_0, x_1, \cdots, x_{n-1}\}$, each taking binary values of 0 (exclusion) or 1 (inclusion). The objective is to select and recommend a configuration of indexes that maximizes the performance benefit for the workload while adhering to the storage/validity constraints.

3.1 Notation

The various input and algorithmic parameters used in the sequel, together with their notation, are summarized in Table 1.

Table 1: Parameters

Type	Symbol	Description	Domain
Input	I	List of Indexes	$[i_0, \cdots, i_{n-1}]$
Input	W	List of Storage Costs	$[w_0, \cdots, w_{n-1}]$
Input	V	List of Time Benefits	$[v_0, \cdots, v_{n-1}]$
Input	W_{max}	Storage Budget	$(0, \sum_{i=0}^{n-1} w_i]$
Input	V_{max}	Maximum Benefit	$[0, \sum_{i=0}^{n-1} v_i]$
Input	L	# of Possible Configurations	$2^{\lvert I \rvert}$
Input	S	# of Shots	$\mathbb{Z}^+$ [Pos. Integer]
OQIA	p	Repetition Depth of QAOA	$\{1, \cdots, \log(\lvert I \rvert)\}$
SQIA	δ	Desired Optimality Probability	$[0.5, 1)$
SQIA	ϵ	Failure Probability	$1 - \delta$
SQIA	λ	Step size for GGS	$[1, 1.33]$
SQIA	α	Timeout Control of GGS	$\mathbb{Q}^+$ [Pos. Rational]
SQIA	Max_{iter}	Upper bound on Grover Iteration	$\lfloor \alpha \cdot \sqrt{L} \rfloor$
SQIA	R	# of Repetitions of GGS	$\lceil \log(1/\epsilon) \rceil$

In the subsequent sections, we describe the proposed Optimization-based (OQIA) and Search-based (SQIA) quantum schemes to implement Stage 4 of the Index Advisor pipeline.

4 OPTIMIZATION-BASED QIA (OQIA)

In the first step of OQIA, the index selection optimization problem, as defined in Equation 2, is transformed into a Quadratic Unconstrained Binary Optimization (QUBO) instance. We then map the QUBO instance to an Ising Hamiltonian using the transformations outlined in [40]. This two-step process helps convert the index selection problem instance into a suitable format for consumption by Quantum Approximate Optimization Algorithm (QAOA).

QUBO for Index Selection. QUBO problems feature binary decision variables, quadratic objective functions, and no constraints – the objective is to identify the assignment of binary variables that minimizes the quadratic objective function. For the index selection problem, we essentially use the QUBO reformulation technique presented in [23] with some minor modifications. The process operates as follows: First, Equation 2 is converted from a maximization task into a minimization task – this is trivially achieved by changing the sign of the objective function:

$$\sum_{i=0}^{n-1} x_i v_i \rightarrow -\sum_{i=0}^{n-1} x_i v_i \tag{3}$$

Next, the storage space constraint is internalized to make the optimization objective constraint-free. This process requires additional machinery, specifically the introduction of an auxiliary term, denoted $\mathcal{B}$ in the optimization objective. This $\mathcal{B}$ term undergoes dynamic adjustments in each iteration by the optimizer in response to the solution's characteristics.

For starters, the original solution array X of length n bits is extended to include an additional m bits, resulting in an expanded array denoted $X^B = [x_0, \cdots, x_{n-1}, b_n, \cdots, b_{n+m-1}]$. The additional bits b_j are formed from the binary representation of $\mathcal{B}$, that is, $\mathcal{B} = \sum_{j=n}^{n+m-1} 2^j b_j$.

Next, the objective function is updated to enable the optimizer to simultaneously optimize the values of X and $\mathcal{B}$. To do so, the $\mathcal{B}$ is subtracted from the storage constraint, the resultant is squared,

and then multiplied with a large positive number A. The resulting expression is:

$$A \cdot \left(W_{max} - \sum_{i=0}^{n-1} x_i w_i - \mathcal{B} \right)^2 \tag{4}$$

The updated cost function is obtained by adding Equations 3 and 4:

$$\begin{aligned} C(X^B) &= A \cdot \left(W_{max} - \sum_{i=0}^{n-1} x_i w_i - \mathcal{B} \right)^2 - \sum_{i=0}^{n-1} x_i v_i \\ &= A \cdot \left(W_{max} - \sum_{i=0}^{n-1} x_i w_i - \sum_{j=n}^{n+m-1} 2^j b_j \right)^2 - \sum_{i=0}^{n-1} x_i v_i \end{aligned} \tag{5}$$

resulting in the following QUBO objective function:

$$\min_{X^B} \left(A \cdot \left(W_{max} - \sum_{i=0}^{n-1} x_i w_i - \sum_{j=n}^{n+m-1} 2^j b_j \right)^2 - \sum_{i=0}^{n-1} x_i v_i \right) \tag{6}$$

The minimization of the above function requires the first term to go to 0, implying that $\sum_i x_i w_i \leq W_{max}$, which is the storage constraint. Given a zero-valued first term, the function minimization is determined by the second term, which is the aggregate benefit. Therefore, benefit maximization is achieved subject to meeting the storage budget. Finally, we map the generated QUBO instance (Equation 6) to an Ising Hamiltonian using the scheme in [40].

4.1 OQIA Resource Scaling

The number of qubits required to implement the OQIA pipeline is determined by the count of binary variables in the QUBO objective function. As shown in Equation 6, it comprises two sets of binary variables: (a) the n binary variables $\{x_0 \cdots x_{n-1}\}$ representing indexes, and (b) the m binary variables $\{b_n, \cdots, b_{n+m-1}\}$ composing $\mathcal{B}$ which are upper-bounded by the storage budget W_{max}. Assuming that W_{max} fits within a 32-bit integer, the qubit count scales *linearly* with the number of indexes.

5 SEARCH-BASED QIA (SQIA)

We now turn our attention to solving the index selection problem as an enumerative search over the exponential configuration space using the Grover Search (GS) algorithm. The quantum *superposition* property is first leveraged to generate the exponential space of candidate index configurations, incurring only logarithmic qubit overhead. Next, we present a novel algorithm for constructing a quantum oracle for the index selection problem. Our oracle harnesses the power of quantum *entanglement* and, for reasons explained below, loads the problem instance in the qubit *phases* – a marked shift from the normal practice of loading data into qubit basis states. Further, only standard quantum gates are used in our construction of the oracle. Then, we provide procedures for configuring the GS algorithm to overcome various practical challenges. Finally, we analyze the computational complexity of our approach and demonstrate that SQIA effectively provides *linear* qubit scalability with index set size while preserving the quadratic speed-up offered by the GS algorithm.

Generate Candidate Configurations. Given the set of candidate indexes I, the total number of possible candidate configurations is $L = 2^{|I|}$. The *superposition* property is used to simultaneously load these L candidate configurations in I qubits, employing a corresponding number of **H** gates.

To achieve this, a quantum circuit is initialized with $n = |I|$ qubits, where each qubit maps to an element of the index set I. Initially, all the qubits are in the $|0\rangle$ quantum state. In order to implicitly generate the power set of I, we apply the **H** gate *individually* on all qubits, and then *observe the combined state* of these qubits. Accordingly, we get:

$$\begin{aligned} H|0\rangle_0 \otimes \cdots \otimes H|0\rangle_{|I|-1} &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\ &= \frac{1}{\sqrt{L}} \sum_{x=0}^{L-1} |x\rangle \end{aligned} \tag{7}$$

Since $|x\rangle$ corresponds to the binary representation of the corresponding integer, the above equation represents an equal superposition of all L candidate index configurations. To visualize, consider the binary representation of any integer $x \in \{0, L-1\}$. For every bit $j \in \{1, |I|\}$, if $x_j = 1$, then include the index i_j in the corresponding candidate configuration, otherwise not.

5.1 Quantum Oracle for Index Selection

We now move on to showing how the quantum *entanglement* property can be leveraged for assigning storage costs and time benefits to the candidate configurations, and maintaining compliance with the storage constraint. This is done in conjunction with qubit phase manipulation. We present novel building blocks that construct a quantum oracle for the index selection problem, and this oracle is subsequently used in the GS algorithm to identify the qualifying configurations.

5.1.1 **Encoding Storage Costs (Index Weights).** Given a candidate configuration C_x, $W(C_x)$ represents the aggregate cost of its constituent indexes. For instance, if $C_x = \{i_0, i_2\}$, then its cost $W(C_x) = w_0 + w_2$. Further, an upper-bound on this value is the cost of the configuration that includes all candidate indexes. Therefore, we need $m = \lceil \log_2(\sum_{i=0}^{|I|-1} w_i) \rceil$ qubits to cover all costs that could potentially appear during execution.

Direct Approach. The simplest way to associate costs to candidate configurations is to first pre-compute the costs of all candidate configurations. Then, to single out each configuration (present in the uniform superposition) using the quantum **X** gate, and insert the corresponding cost in the dedicated cost qubits. But this requires: 1) Pre-computing costs for an exponential number of configurations; and 2) Applying an exponential number of **X** gates to uniquely identify each candidate configuration.

Phase-based Approach. Given the above problems with the direct approach, we design an alternate strategy based on qubit *phase* manipulation. The state of a qubit, as shown by Equation 1, has three components: the basis states ($|0\rangle$ and $|1\rangle$), the amplitudes (α and β) associated with the basis states , and the relative phase (γ) of the $|1\rangle$ state. We convert the classical costs into suitable angles γ (in Fourier basis), and then load them as the relative phase of the qubits

using the **CP** gate. This strategy helps us to intrinsically compute and associate the costs of an exponential number of candidate configurations using just $m \cdot |I|$ two-qubit **CP** gates.

Algorithm 1 load_cost_SQIA

Require: $W = [w_0, w_1, \cdots, w_{n-1}]$ ▷ Input costs
1: $n \leftarrow |W|$ ▷ # of Indexes
2: $m = \lceil \log_2(\sum_{i=0}^{n-1} w_i) \rceil + 1$
3: $qc \leftarrow QuantumCircuit(n+m)$
4: $qc.h(m)$ ▷ Generate Equal Superposition
5: **for** $i \in range(n)$ **do**
6: $\quad \theta_i = \frac{w_i}{2^m} \cdot 2\pi$
7: $\quad$ **for** $j \in range(m)$ **do**
8: $\quad\quad \gamma_j = 2^{(m-j-1)} \cdot \theta_i$
9: $\quad\quad qc.cp(\gamma_j, i, j)$
10: **return** qc

The above process is summarized in Algorithm 1, which takes a list of costs W as input. As mentioned earlier, we need m qubits to encode the costs of all candidate configurations. However, an additional qubit is allocated (Line 2) for storing the *sign* of the costs. Specifically, a state $|0\rangle$ in the sign qubit represents a positive cost. Subsequently, we instantiate a quantum circuit qc with the capacity to accommodate both index (n) and cost ($m+1$) qubits – all of these qubits are initialized to the $|+\rangle$ state using the **H** gate.

Now consider an index $i \in I$ with cost w_i. To load this cost into the m cost qubits, the phase angles $\gamma_1, \cdots, \gamma_m$ are computed for each cost qubit. For ease of understanding, we decompose this angle computation into two parts:

Rotation Angle: The core rotation angle θ_i is the rotation relative to the full rotation 2π. It depends on w_i and the number of qubits, namely m, on which this cost is encoded and calculated as $\theta_i = \frac{2*\pi}{2^m} \cdot w_i$

Rotation Frequency: The frequency of rotation depends on the qubit position – the first qubit is rotated by an angle $2^{m-1} \cdot \theta_i$, and this angle is progressively halved for the next qubits, i.e., the j^{th} qubit is rotated by an angle $\gamma_j = 2^{m-j-1} \cdot \theta_i$.

The rotation γ_j is applied using the **CP** gate. Here, the index qubit i is the control qubit and the j^{th} cost qubit is the target. The **CP** gate *entangles* the index qubits with the cost qubits, and γ_i loads the costs in the phase of these qubits. We repeat this process over all indexes in I.

Phase Loading Example. To make clear the above process of loading costs via phases, consider the candidate configuration $C_x = \{i_0, i_2\}$ for the problem instance shown in Figure 1. Here, the aggregate cost $W(C_x) = w_0 + w_2$, and the index qubit assumes the quantum state $|0000101\rangle$ (in superposition). It therefore triggers the conditional rotations corresponding to index qubits i_0 and i_2. The index qubit i_0 adds a relative phase γ_j to the j^{th} cost qubit (m_j), computed as:

$$\gamma_j = 2^{(m-j-1)} \cdot 2\pi \cdot \frac{w_0}{2^m} \tag{8}$$

producing a quantum state $|m_j\rangle = \frac{|0\rangle + e^{i \cdot 2^{-(j+1)} \cdot 2\pi \cdot w_0} |1\rangle}{2}$. Similarly, index qubit i_2 adds a phase angle $\gamma_j = 2^{-(j+1)} \cdot 2\pi \cdot w_2$ to m_j. Therefore, the final state of m_j is:

$$|m_j\rangle = \frac{|0\rangle + e^{i \cdot 2^{-(j+1)} \cdot 2\pi \cdot (w_0 + w_2)} |1\rangle}{2} \tag{9}$$

With the above rotations, the desired aggregate cost is loaded in the phase of the cost qubits.

5.1.2 **Encoding Benefits.** The encoding of benefits mirrors the encoding of costs, and Algorithm 1 is reused with minor modifications. Specifically, the list of benefits V is passed instead of the costs W, and the number of qubits needed is $v = \lceil \log_2(\sum_{i=0}^{n-1} v_i) \rceil + 1$. The rest of the algorithm is followed as is.

5.1.3 **Encoding Storage Constraint.** To encode the storage constraint, W_{max} is subtracted from the cost qubits for all L configurations in superposition. For this, we encode the negative of the storage budget ($-1 \cdot W_{max}$) as angles ($\gamma_1, \cdots, \gamma_m$), and apply a single-qubit **Phase (P)** gate on each cost qubit. The **P** gate is used instead of the **CP** gate to make the rotation independent of the index qubits, thus uniformly subtracting the storage constraint from the cost of an exponential number of superposed costs with only one application of the gate. After this step, all configurations that satisfy the storage constraint will have a negative cost loaded as a relative phase in their cost qubits.

5.1.4 **Extracting Signed Costs.** As discussed in Section 2, relative phases are not directly measurable. To make them measurable, we apply the *inverse Quantum Fourier Transform* (QFT) algorithm [43] to the cost qubits and transform the costs from the phase to the basis state of the qubits. A key point to note here is that since the costs are encoded in the angles of a periodic function (e^{iY}) with a period 2π, when the inverse QFT operation is performed, the costs are obtained in *two's complement* format. Accordingly, costs associated with the configurations that satisfy the storage constraint are all either 0 or negative, and negative configurations can be easily identified since their sign qubit is in the quantum state $|1\rangle$. Further, the 0 cost configurations are identified by ignoring the sign qubit and checking if the rest of the cost qubits are in the $|0\rangle$ state. These checks are easily encoded in the quantum circuit using **MCT** gates to detect and signal qualifying configurations.

5.1.5 **Composing the Quantum Oracle.** For a given instance of the index selection problem, we compose the above modules (cost encoding, benefit encoding, constraint encoding, cost extraction), combine their circuits and construct the quantum oracle. The oracle integrates with the GS algorithm and signals the qualifying configurations. The full construction details are available in [35].

5.1.6 **Oracle Construction Example.** Consider the following index selection problem instance (we will refer to this as **I2**): $I = [i_0, i_1]$, $W = [1, 4]$, $V = [2, 4]$, and $W_{max} = 4$. The quantum oracle for this problem is shown in Figure 5, and the construction process is summarized below. Note, each step below is mapped to the numbers shown in Figure 5, except for the preprocessing step (i.e. Step 0).
Step 0: Count the number of qubits needed to store the indices $n = |I| = 2$, the cost $m = \lceil \log_2(\sum_{i=0}^{n-1} w_i) \rceil + 1 = 4$ and the benefit $v = \lceil \log_2(\sum_{i=0}^{n-1} v_i) \rceil + 1 = 4$. Additionally, one qubit is required for each of the following: encoding the storage constraint, encoding the target benefit constraint, and flagging the qualifying configurations.

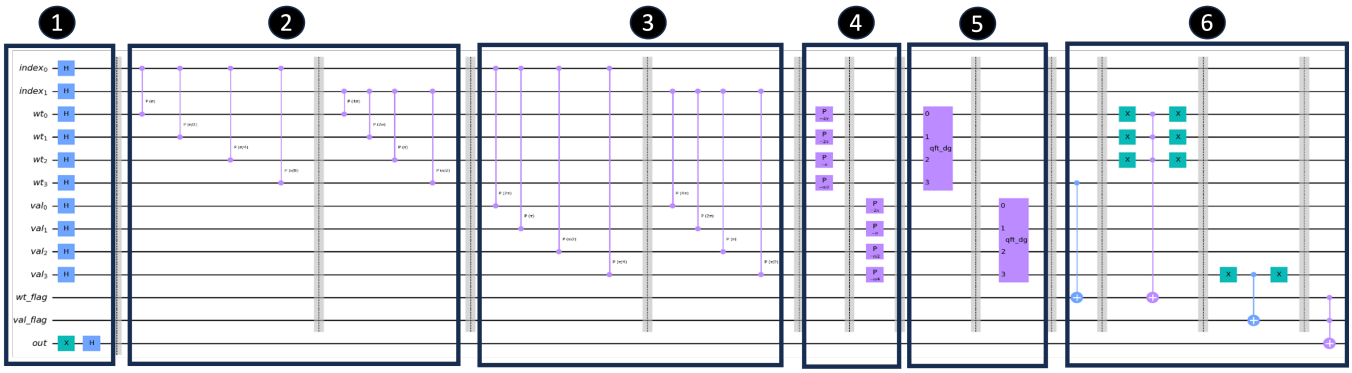


Figure 5: Quantum Circuit of SQIA Oracle for Problem Instance I2

Note, the target benefit constraint is an additional constraint added in SQIA and is explained later in Section 5.2. Therefore, the total number of qubits needed is $2 + 4 + 4 + 3 = 13$.

Step 1: Initialize the index, cost, and benefit qubits in an equal superposition $|+\rangle$ state using the **H** gate and the output qubit in the $|-\rangle$ quantum state. Although the index, cost, and benefit qubits are initialized using the same quantum gate, their interpretations vary depending on how these qubits are used in the rest of the quantum circuit. Specifically, the index qubits are viewed jointly; hence, as shown earlier, they generate all the 4 possible index configurations after initialization. The remaining qubits are viewed independently – the cost and benefit qubits are kept in state $|+\rangle$ to load and process the data in the relative phase of the qubit while the output qubit is in state $|-\rangle$ to flag the qualifying index configurations.

Step 2: For each $w_i \in W$, the phase angles are calculated and loaded into the relative phase of the cost qubits by applying the **CP** gate. For example, for $w_0 = 1$, the rotation angle $\theta_0 = \frac{2*\pi}{2^m} \cdot w_0 = \frac{\pi}{8}$ and therefore the phase angles are $\gamma_0 = \pi, \gamma_1 = \frac{\pi}{2}$, $\gamma_2 = \frac{\pi}{4}$ *and* $\gamma_3 = \frac{\pi}{8}$.

Step 3: Similarly, for each benefit $v_i \in V$, the phase angles are computed and loaded into the relative phase of the benefit qubits.

Step 4: Next, the storage and the target benefit constraints are applied. For $W_{max} = 4$, the phase angle is computed for -4 and applied to all cost qubits using a quantum **P** gate.

Step 5: Now, to retrieve the signed cost and benefit values, the inverse QFT operation is performed on the cost and benefit qubits.

Step 6: Finally, the qualifying index configurations are signalled via the output qubit.

5.2 Finding Optimal Index Configuration

Now we turn our attention to the index selection process, implemented via the **SQIA _search** procedure outlined in Algorithm 2. It takes a problem instance comprising $[I, W, V, W_{max}]$ as input, and produces an optimal index configuration with a user-settable success probability $\delta \in [0.5, 1)$.

Since the precise benefit accrued by the optimal configuration is initially unknown, Algorithm 2 employs a recursive halving strategy, starting with V_{max}, the upper bound, to identify this value. In each iteration, the $find\_opt\_config$ procedure is used to identify the optimal index configuration achieving a benefit greater than or equal to the target benefit value. Failure indicates that the target benefit is too large to be feasible. The first successful identification indicates a transition from an infeasible to a feasible range, and we now again carry out a recursive halving within this transition range to finally identify the optimal configuration.

At its core, the $find\_opt\_config$ method relies on the GS algorithm. However, as mentioned in Section 2, for effective utilization of the GS algorithm, three key elements must be provided: 1) an appropriate quantum oracle, 2) the precise number of Grover Iterations, and 3) an enhancement of its success probability from 0.5 to the desired δ. While Section 5.1 handled the first element, the following discussion will address the remaining elements.

Algorithm 2 SQIA _search

Require: List: (I, W, V), Int: W_{max}

$V_{target} = V_{max}$
$V_{min} = 0$
$opt_{ind} = null$
while $V_{min} < V_{target}$ **do**
  $P_{res} \leftarrow find\_opt\_config(I, W, V, W_{max}, V_{target}, \delta)$
  if $P_{res}[qualify] == True$ **then**
    $opt_{ind} = P_{res}[ind]$
    $V_{min} = V_{target}$
    $V_{target} = \lceil (P_{res}[val] + V_{max})/2 \rceil$
  else
    $V_{target} = \lceil (V_{min} + V_{target})/2 \rceil$
return opt_{ind}

5.2.1 **Finding precise number of Grover Iterations.** To calculate the number of Grover Iterations (Oracle + Diffusion), the GS algorithm needs precise knowledge of the number of qualifying index configurations. However, since this information is unavailable, we turn to the Generalized Grover Search (GGS) algorithm [11]. In one GGS run, the GS quantum circuit is executed iteratively, with the number of iterations dynamically adjusted in each instance until success. In our work, we have implemented the time-out variant of GGS, with a fixed maximum budget of iterations denoted as Max_{iter}. Furthermore, in each iteration, the number of iterations j is uniformly sampled from the range $[1, l]$, where l is initially set to a constant λ. Subsequently, l is incremented by an amount λ after each iteration. The rationale behind choosing λ is detailed in [14].

Setting Max_{iter}. In the worst case, the GS algorithm needs $\sqrt{L}$ iterations. Therefore, the iteration budget for GGS is set to $\lfloor \alpha \cdot \sqrt{L} \rfloor$, where α is a positive rational number. A universal *lower bound* for α, specifically $\alpha_{lb} = 9.2$, was presented in [14]. In Section 6.7.1 of the technical report [35], we discuss instance-specific *upper bounds*.

5.2.2 **Boosting Success Probability.** The GGS algorithm finds a valid solution with probability 0.5. To boost the probability to the desired δ, we employ the Powering Lemma [34], which resorts to repeat executions of every GGS run. The number of repetitions required for the boosting is $R = \lceil \log(\frac{1}{1-\delta}) \rceil$.

The above considerations are incorporated into the $find\_opt\_config$ procedure, detailed in [35].

5.3 Computational Complexity

Aligned with the quantum computing literature, our complexity measure is the number of calls made to the quantum oracle, since it is architecture-independent. The SQIA _search procedure (Algorithm 2) performs recursive halving over the range $[0, V_{max}]$ which takes a maximum of $\lceil \log_2(V_{max}) \rceil$ steps. Now, for every target benefit value V_{target}, the algorithm invokes the GGS algorithm $R = \lceil \log \left(\frac{1}{1-\delta} \right) \rceil$ times. In each run of GGS, we execute the GS algorithm multiple times but with a budget of Max_{iter} iterations, which is heuristically set to $\lfloor \alpha \cdot \sqrt{L} \rfloor$. Further, each iteration makes 2 invocations of the quantum oracle, once to apply the Oracle, and the second time to undo its effect. Therefore, the total number of Oracle calls made by Algorithm 2 is upper bounded by:

$$\lceil 2 \cdot \lfloor \alpha \cdot \sqrt{L} \rfloor \cdot \lceil \log \left(\frac{1}{1-\delta} \right) \rceil \cdot \lceil \log_2(V_{max}) \rceil \rceil \tag{10}$$

As mentioned earlier, a universal lower bound $\alpha_{lb} = 9.2$ for all L, and the success probability δ is set by the user during system initialization. Therefore, only the terms $\sqrt{L}$ and $V_{\max}$ vary based on the input problem instance. Furthermore, if we reasonably assume that $V_{\max}$ can fit within a 32-bit integer, then $\log_2(V_{max})$ can be upper bounded to 32. Therefore, we can conclude that the overall rate of growth of the number of oracle calls for SQIA is $O(\sqrt{L})$. In contrast, for exhaustive search, the rate of growth of the number of oracle calls is $O(L)$. Consequently, as the problem size increases, the absolute gap between the two approaches widens, resulting in greater time benefits with SQIA for the larger problem instances typically encountered in industrial scenarios.

5.4 SQIA Resource Scaling

The number of qubits required to implement the SQIA pipeline is detailed in Table 2. The entries in the table show that the qubit count exhibits a *linear* relationship with the number of indexes and a *logarithmic* correlation with the cumulative cost and benefits. However, if we reasonably assume that the aggregate cost & benefit (log terms) can fit within a 32-bit integer, in that case the qubit count effectively scales *linearly* with the number of indexes.

6 EXPERIMENTS

In principle, the correct methodology for evaluating quantum performance relative to the classical approaches would be to execute both classes of algorithms on their respective devices, assess the quality of the recommended outcomes, and measure the index selection overheads. However, this is not practical at the current time due to lack of industrial-strength quantum platforms. Therefore, we settle for the approach prevalent in the quantum computing literature (e.g. [14, 30]), wherein comparisons are on architecture-independent metrics – in our case, the number of oracle calls.

Table 2: SQIA Qubit Requirement

Symbol	Description	Count
n	# of Qubits for Indexes	$\lvert I \rvert$
m	# of Qubits for Cost	$\lceil \log_2(\sum_{i=0}^{n-1} w_i) \rceil + 1$
v	# of Qubits for Benefit	$\lceil \log_2(\sum_{i=0}^{n-1} v_i) \rceil + 1$
c	# of Qubits for Constraints	2
out	# of Qubits for Output	1

6.1 Experiment Environment

We implemented the proposed ideas using the Qiskit SDK [33] and performed evaluations with Qiskit Aer [3], on a **32-qubit** gate-based noiseless simulator. Using Qiskit Runtime Primitives [4], the same code was also ported to and evaluated on an IBM Eagle circuit processor with 127 qubits ("ibm_sherbrooke"). Our experiments have perforce been carried out on modest problem instances due to current platform limitations; however, we expect the design techniques to carry through to futuristic scaled platforms.

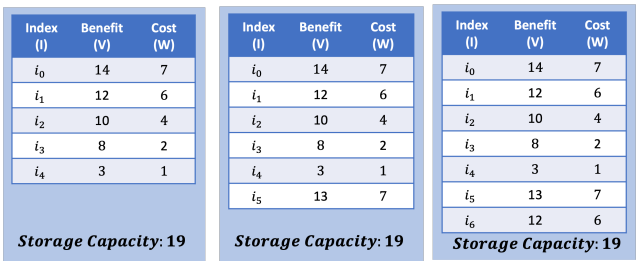
As shown in Stage 4 of Figure 2, the Quantum Index Advisor module receives an index selection problem instance comprising of (a) an index set (I), (b) the associated time benefit (V) and storage cost (W) of each index in I, and (c) the storage budget W_{max}. In our evaluation, the instance is solved with the proposed quantum schemes, OQIA and SQIA, and compared with the classical baselines, Greedy and Exhaustive Search.

Our problem suite consists of four index selection instances. The first instance, comprising 7 indexes, is the motivating example of Figure 1, generated on the commercial database engine – we refer to it as **CDB_I7**. The remaining three problem instances comprising 5, 6, and 7 indices, respectively, are synthetically generated – we hereafter refer to them as **I5**, **I6**, and **I7**. These problem instances are shown in Figure 6, and they all have the same storage constraint, namely $W_{max} = 19$. In addition, in all of them, Exhaustive Search provides the same optimal configuration, namely $\{i_0, i_1, i_2, i_3\}$ with benefit 44, while Greedy provides the same (sub-optimal) recommendation, namely $\{i_0, i_2, i_3, i_4\}$ with benefit 35. We define the quality of a configuration as its benefit normalized to the ideal solution (as obtained by the Exhaustive Search algorithm).

While the original CDB_I7 instance could be directly used with OQIA, its costs and benefits were normalized for SQIA evaluation to reduce the complexity of the quantum circuit. Specifically, the following transformed problem instance produces the same greedy and optimal solution as the original problem: $I = [i_0, i_1, i_2, i_3, i_4, i_5, i_6]$, $W = [126, 114, 3, 72, 95, 1, 4]$, $V = [4, 5, 27, 27, 27, 1, 1]$, and $W_{max} = 75$. Furthermore, the algorithmic parameters for OQIA were set to ($p = 1, S = 100$), while SQIA had ($\delta = 0.9, S = 1$). The sensitivity to these parameters is discussed later in the section.

Table 3: Evaluation of IA Schemes on a 32-qubit Quantum Simulator

Problem	# of Candidate Configurations	IA Scheme	Configuration Quality			Quantum Resources		Normalized Overhead
			Weighted Average	Optimal Fraction	Worst Case	Qubits	Depth	
CDB_I7	128	Exhaustive	1.0	1.0	1.0	–		100%
		SQIA (δ = 0.9, α = 0.26, S = 1)	0.95	0.9	0.54	28	80	73%
		OQIA (p = 1, S = 100)	0.76	0.5	0.5	15	30	23%
		Greedy	0.54	0	0.54	–		–
I5	32	Exhaustive	1.0	1.0	1.0	–		100%
		SQIA (δ = 0.9, α = 0.18, S = 1)	0.9	0.8	0.34	21	58	111%
		OQIA (p = 1, S = 100)	0.99	0.9	0.89	10	20	91%
		Greedy	0.8	0	0.8	–		–
I6	64	Exhaustive	1.0	1.0	1.0	–		100%
		SQIA (δ = 0.9, α = 0.22, S = 1)	1.0	1.0	1.0	22	60	94%
		OQIA (p = 1, S = 100)	0.97	0.6	0.86	11	22	45%
		Greedy	0.8	0	0.8	–		–
I7	128	Exhaustive	1.0	1.0	1.0	–		100%
		SQIA (δ = 0.9, α = 0.26, S = 1)	1.0	1.0	1.0	25	68	75%
		OQIA (p = 1, S = 100)	0.97	0.4	0.89	12	24	23%
		Greedy	0.8	0	0.8	–		–



(a) I5 **(b) I6** **(c) I7**

Figure 6: Index Selection Problem Suite

6.2 Performance Comparison

Configuration Quality. Table 3 presents a summary assessment of the configuration quality delivered by the four index selection strategies when invoked on our problem suite on the noiseless quantum simulator. For the quantum algorithms, the Weighted Average column reports the average of the quality scores over the ten repeat invocations, the Optimal Fraction column represents the fraction of outcomes delivering the optimal benefit, while the Worst Case column represents the smallest benefit obtained across the invocations. Recall that the expectation from OQIA is to recommend a solution having better quality than the Greedy scheme, while SQIA should produce the optimal solution with the desired δ probability. The good news from these results is that both schemes consistently achieve their objectives (except for I5). The detailed instance-specific analysis is as follows:

CDB_I7: The Greedy configuration delivers only 0.54 of the optimal. OQIA improves the quality to 0.76, while SQIA delivers the optimal configuration with the desired δ = 90%. Nevertheless, owing to the inherently probabilistic nature of quantum platforms, the worst-case recommendation could be of arbitrary quality. However, as observed for both schemes, the worst-case quality is about the same as the greedy solution.

I5: Here, Greedy delivers a configuration quality of 0.8, and OQIA enhances the configuration quality to as high as 0.99. On the other hand, SQIA although delivering 0.9 quality, does not satisfy its desired δ success probability. This is because the α value is impractical for this instance, as explained in detail in the technical report [35].

I6 and I7: In both these instances, OQIA enhances the solution quality to 0.97. Further, SQIA always recommends the optimal solution, and exceeds the δ threshold. The worst-case quality of both schemes is also significantly better than the greedy recommendation. Notably, this exceptional performance is delivered despite the exponential increase in the number of candidate configurations from **I6** to **I7**.

Computational Overheads. Turning our attention to the computational effort, also delineated in Table 3, we observe that for **CDB_I7**, **I6** and **I7**, SQIA incurs only marginally fewer Oracle calls compared to Exhaustive Search. This is further substantiated when we consider the smallest-sized **I5**, where the Oracle calls even exceed those of Exhaustive Search. This may seem surprising; however, this is an artifact of our small-sized examples and is again due to the impractical values of α. The resource gap will become clearly apparent in large-index scenarios seen in enterprise environments. For instance, consider the full TPC-H benchmark query suite with 53 single-attribute candidate indexes [37]. Assuming that the aggregate benefits can be accommodated in a 32-bit integer, then for δ = 0.9, we estimate using Equation 10 that SQIA will only need 0.002% of the oracle calls incurred by Exhaustive Search.

A similar trend is observed for the OQIA scheme and is attributed to the variational principle used by the underlying QAOA algorithm, which dynamically explores and refines the solution space, and quickly converges to optimal or near-optimal solutions. Hence, it is evident that both the proposed schemes are targeted towards larger problem instances. In the technical report [35], we delineate the problem size landscape in which the SQIA scheme ensures both guaranteed quality and computational advantage.

6.3 Real Circuit Processor

We now turn our attention to the performance observed on a popular real quantum circuit processor, the 127-qubit Eagle from IBM. To optimize the quantum circuits for this hardware and reduce the impact of errors, we used Qiskit runtime compilation techniques and transpiled the Qiskit simulator quantum circuits for OQIA and SQIA by setting the optimization level to 3 [6] and verified that they ran correctly. Further, their performance on the index instances of Table 3 was evaluated and the results are summarized in Table 4. The Configuration Quality column reports the average of the quality scores over three repeat invocations. The outcomes of these experiments are qualitatively in agreement with the simulated results, demonstrating that achieving high-quality solutions is feasible even on these early quantum hardware. Moreover, an additional insight is that OQIA is more robust to quantum noise than SQIA, leading to better configurations on this platform. This is due to the QAOA algorithm used in OQIA, which is more effective in noisy environments than the GS algorithm employed by SQIA.

6.4 Real Annealing Processor

For completeness, we have also evaluated the performance of index selection on a real annealing processor, specifically the 5000-qubit D-Wave Leap Hybrid Solver [1]. In these experiments, the QUBO generated in OQIA was input to this solver. Given the native suitability of annealing for optimization, this approach provided, as expected, the best solutions for all the small-sized index selection instances considered in our study.

However, this success needs to be qualified with the following deployment-related observations: (1) Extending the index selection problem to account for index interactions requires benefit values to be dynamically computed, as they depend on the order in which the indexes are selected. Encoding such dynamism in the QUBO may not be feasible, whereas a circuit processor offers the necessary computational flexibility. (2) The overall industry trend is towards circuit processors, with even D-Wave itself recently including such processors in its roadmap [39]; (3) In a practical DBMS, it appears reasonable to expect a *single* quantum platform that is usable for *both* computation and optimization. Therefore, hosting index selection on circuit processors is of independent interest.

Table 4: Evaluation on IBM 127-qubit Eagle Processor

Problem	IA Scheme	Configuration Quality
CDB_I7	SQIA (δ = 0.9, α = 0.26, S = 1)	0.85
	OQIA (p = 1, S = 100)	0.68
I5	SQIA (δ = 0.9, α = 0.18, S = 1)	0.89
	OQIA (p = 1, S = 100)	1.0
I6	SQIA (δ = 0.9, α = 0.22, S = 1)	0.86
	OQIA (p = 1, S = 100)	0.98
I7	SQIA (δ = 0.9, α = 0.26, S = 1)	0.95
	OQIA (p = 1, S = 100)	1.0

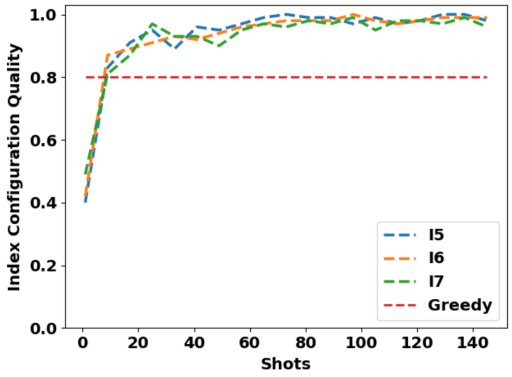


Figure 7: OQIA Solution Quality vs Shots ($p = 1$)

6.5 Discussion

6.5.1 **Setting quantum parameters.** We have evaluated the sensitivity of the quantum algorithm performance to the various configuration parameters. Due to space constraints, the full details are deferred to the technical report (Sections 6.6 and 6.7) [35]. Here, as an example, we discuss choosing the number of shots for OQIA.

To find the optimal number of shots, we evaluated OQIA for the I5, I6, and I7 problems, setting $p = 1$ and varying S in the range $[1, 150]$. Each experiment was repeated ten times and Figure 7 shows the average quality score against S. For comparative purposes, the performances of Greedy heuristic is also shown. Three key insights emerge from this figure: 1) OQIA rapidly outperforms the greedy algorithm after S crosses a small value ($\geq$ 10); 2) The configuration quality achieved by OQIA consistently surpasses that of the greedy solution, resulting in a superior approximation ratio; and 3) Starting from around 100 shots, the recommended configuration is effectively optimal.

6.5.2 **Estimated Efficiency on Practical Workloads.** We now project the performance profile that could be expected on the full TPC-H benchmark query suite. The number of candidate single-attribute indexes is 53 [37]. Assume that the aggregate costs and benefits can be accommodated in 32-bit integers, and that the estimated depth of the Quantum Oracle circuit constructed in the SQIA scheme is around 100 (calculated by analyzing Algorithm 2). Furthermore, as shown in [20], a single two-qubit gate currently takes around 6.5ns. Now, anticipating a reduction to 1ns within the next decade, a quantum Oracle call in the SQIA scheme is estimated to take around 100ns. Next, assuming $\delta = 0.9$, we can use Equation 10 to estimate the number of Oracle calls made by the SQIA scheme. Multiplying this by the time for an Oracle call, the SQIA scheme is estimated to take approximately 5 hours to identify an *optimal* configuration with 90% probability. In contrast, assuming a classical Oracle call duration of just 1ns, an Exhaustive Search would take around 3.5 *months* to find the optimal solution.

7 RELATED WORK

Recently, there have been vision papers advocating the need to accelerate database tasks using quantum computing [15, 31, 56, 57]. But, we are not aware of any prior work performing index selection

using quantum platforms. Therefore, in this section, we separately review the literature on index selection, 0-1 Knapsack Problem on quantum platform, and use of quantum platforms for DBMS.

Index Selection. The index selection problem has been studied for decades, and recent comprehensive surveys are available in [32, 37]. Further, all major database engines feature an Index Advisor. We have already considered DB2's Index Advisor in the preceding sections. Microsoft SQL Server features a broad-based Database Engine Tuning Advisor (DTA) [18], which includes a sophisticated index advisor in its ambit. DTA considers both single and multi-column indexes, as well as their interactions.

Recently, ML based IA methods have also been introduced [36, 38]. Most of this work uses reinforcement learning, where the state is defined as the currently built indexes, and the action as choosing an index to build. These methods exhibit promising outcomes in enhancing the index selection process's efficiency. However, as demonstrated in the detailed evaluation of [36], the quality of the recommended configuration remains similar to non-ML systems.

The approaches proposed in our work aim to enhance the configuration quality provided by the above tools by leveraging the computing power offered by quantum platforms.

0-1 Knapsack Problem on quantum platform. The papers in this area could be broadly classified into two categories: Some address a weaker formulation of the original problem, while others use non-standard quantum gates with heuristic parameter settings. Specifically in [26], the authors consider 0-1 Knapsack Problem instances that do not have item-specific benefit values. Whereas in [54], an approach called "Quantum Tree Generator (QTG)" was introduced. Here, they generate in superposition all feasible solutions for a given problem instance and then leverage the Grover Search algorithm to find the solution. However, their protocol uses non-standard biased Hadamard gates and heuristically sets the bias value. These deviations raise concerns about the feasibility of implemention on real quantum computers. In contrast, we have considered standard 0-1 Knapsack Problem and utilized only standard quantum gates.

Quantum Database Platforms. There have been some earlier efforts to showcase the potential of quantum platforms for database optimization. For instance, the generation of optimal execution plans in the context of Multi-Query Optimization was proposed in the pioneering work of [51]. Their technique is based on utilizing the quantum annealing. Moreover, a singular feature of the study is its implementation on the D-Wave Quantum Annealer [2].

More recently, the relational join-order optimization problem was addressed in [42, 49, 55] on quantum hardware. These proposals reformulated the problem to an equivalent QUBO task that can be evaluated on quantum computers. Specifically, [49] conducted a comprehensive evaluation of various query graphs and successfully generated about 41% valid join orders, among which 10% were optimal, for three-relation chain queries using the D-Wave annealing processor. These results demonstrate the feasibility of quantum solutions and also serve as a motivation for our OQIA scheme.

Quantum platforms for database transaction scheduling has also been explored. Specifically, to schedule transactions in a 2PL database, [9, 10] introduced a quantum algorithm that uses annealing , while [29] employed the Grover Search algorithm.

Finally, another line of research focuses on designing "quantum-inspired" algorithms for DBMS [41, 45]. These algorithms are designed to run on classical computers but incorporate ideas derived from quantum computing to potentially improve their performance in solving certain problems. For instance, in [45], the authors perform resource allocation reasoning on traditional relational databases in an OLTP setting. They borrow ideas of quantum superposition and quantum measurement and allow resource transactions to commit without assigning concrete resource instances.

8 CONCLUSIONS AND FUTURE WORK

We presented here, for the first time, Quantum-computing-based Index Advisors for efficiently delivering index selections that provide close-to-optimal benefits under a storage budget. We first described an optimization-based approach, OQIA, which composed well-known quantum algorithms to provide high-quality configurations with limited expense of quantum resources. Then, we designed from scratch a novel Grover Search-based approach SQIA, which provides optimal solutions with high probability, in conjunction with resource consumption that is compatible with the quantum platforms expected in the coming decade. The key novelty was the construction of an efficient quantum oracle where data is represented in qubit phases, rather than basis states, and using only standard quantum gates.

Our design is a hybrid quantum-classical architecture that lends itself to easy implementation on contemporary database environments. Using classical components to enumerate the search space and the benefits and costs of indexes, it leverages the power of the quantum computing platform for the computationally expensive index selection process.

The evaluation of modest index scenarios on both a noiseless quantum simulator and real quantum hardware demonstrated the feasibility of our proposed schemes on quantum platforms. Further, they indicated that high-quality configurations can be reliably produced by suitable choices of algorithmic parameter settings. We also showed that the complexity of our circuit design scales linearly to future deployment scenarios with large databases. In our future work, we plan to evaluate our algorithms on more powerful quantum hardware (eg. 1121 qubit IBM Condor [27]) and also extend our algorithms to include additional IA components.

In summary, we have taken an initial step in this paper toward designing and constructing index advisors using quantum platforms that are both close-to-optimal in solution quality and efficient with regard to index selection. We hope that our results will spur new research to address the challenges of making quantum-based databases a practical reality in the near future.

ACKNOWLEDGMENTS

We are extremely grateful to Anupam Sanghi, Srinivas Karthik, and Kalyan Dasgupta for their constructive feedback on the draft paper. We also thank Sieglinde Pfaendler, SheshaShayee Raghunathan, and Dhinakaran Vinayagamurthy for facilitating access to IBM quantum hardware and warm encouragement of our study. We acknowledge the use of IBM Quantum Credits for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

REFERENCES

[1] D-Wave 2024. *D-Wave Leap's Hybrid Solvers.* D-Wave. Retrieved July 15, 2024 from https://docs.dwavesys.com/docs/latest/doc_leap_hybrid.html

[2] D-Wave 2024. *D-Wave Quantum Annealer.* D-Wave. Retrieved July 15, 2024 from https://www.dwavesys.com/learn/quantum-computing

[3] IBM 2024. *Qiskit Aer.* IBM. Retrieved July 15, 2024 from https://github.com/qiskit/qiskit-aer

[4] IBM 2024. *Qiskit Runtime Primitives.* IBM. Retrieved July 15, 2024 from https://docs.quantum.ibm.com/run/primitives

[5] 2024. Qiskit Tutorials. Retrieved June 10, 2024 from https://www.ibm.com/quantum/qiskit#tutorials

[6] IBM 2024. *Runtime Compilation.* IBM. Retrieved July 15, 2024 from https://docs.quantum.ibm.com/run/configure-runtime-compilation

[7] Sanjay Agrawal, Surajit Chaudhuri, Lubor Kollár, Arunprasad P. Marathe, Vivek R. Narasayya, and Manoj Syamala. 2004. Database Tuning Advisor for Microsoft SQL Server 2005. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, Toronto, Canada, August 31 - September 3 2004.* Morgan Kaufmann, 1110–1121.

[8] Debabrota Basu, Qian Lin, Weidong Chen, Hoang Tam Vo, Zihong Yuan, Pierre Senellart, and Stéphane Bressan. 2015. Cost-Model Oblivious Database Tuning with Reinforcement Learning. In *Database and Expert Systems Applications - 26th International Conference, DEXA 2015, Valencia, Spain, September 1-4, 2015, Proceedings, Part I (Lecture Notes in Computer Science)*, Vol. 9261. Springer, 253–268.

[9] 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.

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

[11] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. 1998. Tight bounds on quantum searching. *Fortschritte der Physik: Progress of Physics* 46, 4-5 (1998), 493–505.

[12] Nicolas Bruno and Surajit Chaudhuri. 2005. Automatic physical database tuning: a relaxation-based approach. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data* (Baltimore, Maryland) *(SIGMOD '05)*. Association for Computing Machinery, New York, NY, USA, 227–238. https://doi.org/10.1145/1066157.1066184

[13] Nicolas Bruno and Surajit Chaudhuri. 2007. An Online Approach to Physical Design Tuning. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007.* IEEE Computer Society, 826–835.

[14] Chris Cade, Marten Folkertsma, Ido Niesen, and Jordi Weggemans. 2023. Quantifying Grover speed-ups beyond asymptotic analysis. *Quantum* 7 (Oct. 2023), 1133.

[15] Umut Çalikyilmaz, Sven Groppe, Jinghua Groppe, Tobias Winker, Stefan Prestel, Farida Shagieva, Daanish Arya, Florian Preis, and Le Gruenwald. 2023. Opportunities for Quantum Acceleration of Databases: Optimization of Queries and Transaction Schedules. *Proc. VLDB Endow.* 16, 9 (may 2023), 2344–2353.

[16] Surajit Chaudhuri and Vivek Narasayya. 1998. AutoAdmin "what-if" index analysis utility. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data* (Seattle, Washington, USA) *(SIGMOD '98)*. Association for Computing Machinery, New York, NY, USA, 367–378.

[17] Surajit Chaudhuri and Vivek Narasayya. 2007. Self-tuning database systems: a decade of progress. In *Proceedings of the 33rd International Conference on Very Large Data Bases* (Vienna, Austria) *(VLDB '07)*. VLDB Endowment, 3–14.

[18] Surajit Chaudhuri and Vivek Narasayya. 2020. Anytime algorithm of database tuning advisor for microsoft sql server.

[19] Surajit Chaudhuri and Vivek R. Narasayya. 1997. An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB '97)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 146–155.

[20] Y. Chew, T. Tomita, T. P. Mahesh, S. Sugawa, S. de Léséleuc, and K. Ohmori. 2022. Ultrafast energy exchange between two single Rydberg atoms on a nanosecond timescale. *Nature Photonics* 16, 10 (Aug. 2022), 724–729.

[21] Benoit Dageville, Dinesh Das, Karl Dias, Khaled Yagoub, Mohamed Zait, and Mohamed Ziauddin. 2004. Automatic SQL tuning in oracle 10g. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30* (Toronto, Canada) *(VLDB '04)*. VLDB Endowment, 1098–1109.

[22] Debabrata Dash, Neoklis Polyzotis, and Anastasia Ailamaki. 2011. CoPhy: a scalable, portable, and interactive index advisor for large workloads. *Proc. VLDB Endow.* 4, 6 (mar 2011), 362–372.

[23] Pierre Dupuy de la Grand'rive and Jean-Francois Hullo. 2019. Knapsack Problem variants of QAOA for battery revenue optimisation. arXiv:1908.02210 https://arxiv.org/abs/1908.02210

[24] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 https://arxiv.org/abs/1411.4028

[25] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2015. A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem. arXiv:1412.6062 https://arxiv.org/abs/1412.6062

[26] Toru Fujimura. 2022. Quantum Algorithm for Knapsack Problem by Matrix Computation with Y-Axis-Rotation (-90 degrees). *Global Journal of Pure and Applied Mathematics Volume 18, Number 2, pp. 511-516* (2022).

[27] Jay Gambetta. 2024. *IBM Quantum Roadmap.* Retrieved July 15, 2024 from https://www.ibm.com/quantum/blog/quantum-roadmap-2033

[28] Jay Gambetta and Matthias Steffen. 2023. *Charting the course to 100,000 qubits.* Retrieved July 15, 2024 from https://research.ibm.com/blog/100k-qubit-supercomputer

[29] 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.

[30] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (Philadelphia, Pennsylvania, USA) *(STOC '96)*. Association for Computing Machinery, New York, NY, USA, 212–219.

[31] Le Gruenwald, Tobias Winker, Umut Çalikyilmaz, Jinghua Groppe, and Sven Groppe. 2023. Index Tuning with Machine Learning on Quantum Computers for Large-Scale Database Applications. 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.

[32] Shiyue Huang, Yanzhao Qin, Xinyi Zhang, Yaofeng Tu, Zhongliang Li, and Bin Cui. 2023. Survey on performance optimization for database systems. *Science China Information Sciences* 66, 2 (2023), 1–23.

[33] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. 2024. Quantum computing with Qiskit. arXiv:2405.08810 https://arxiv.org/abs/2405.08810

[34] Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. 1986. Random generation of combinatorial structures from a uniform distribution. *Theoretical computer science* 43, 2–3 (1986), 169–188.

[35] Manish Kesarwani and Jayant R. Haritsa. 2024. *Index Advisors on Quantum Platforms.* Technical Report. Indian Institute of Science. http://dsl.cds.iisc.ac.in/publications/report/TR/TR-2024-06.pdf

[36] Jan Kossmann, Alexander Kastius, and Rainer Schlosser. 2022. SWIRL: Selection of Workload-aware Indexes using Reinforcement Learning. In *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022.* OpenProceedings.org, 2:155–2:168.

[37] Jan Michael Koßmann. 2023. *Unsupervised database optimization: efficient index selection & data dependency-driven query optimization.* Ph.D. Dissertation. University of Potsdam, Germany. https://publishup.uni-potsdam.de/frontdoor/index/index/docId/58949

[38] Hai Lan, Zhifeng Bao, and Yuwei Peng. 2020. An Index Advisor Using Deep Reinforcement Learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Virtual Event, Ireland) *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 2105–2108.

[39] Frederic Lardinois. 2021. *D-Wave Roadmap.* D-Wave. Retrieved July 15, 2024 from https://techcrunch.com/2021/10/05/d-wave-plans-to-build-a-gate-model-quantum-computer

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

[41] Sayed A. Mohsin, Saad Mohamed Darwish, and Ahmed Younes. 2021. QIACO: A Quantum Dynamic Cost Ant System for Query Optimization in Distributed Database. *IEEE Access* 9 (2021), 15833–15846.

[42] 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* (Seattle, WA, USA) *(BiDEDE '23)*. Association for Computing Machinery, New York, NY, USA, Article 7, 7 pages.

[43] Michael A. Nielsen and Isaac L. Chuang. 2016. *Quantum Computation and Quantum Information (10th Anniversary edition).* Cambridge University Press.

[44] R. Malinga Perera, Bastian Oetomo, Benjamin I. P. Rubinstein, and Renata Borovica-Gajic. 2021. DBA bandits: Self-driving index tuning under ad-hoc, analytical workloads with safety guarantees. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 600–611.

[45] Sudip Roy, Lucja Kot, and Christoph Koch. 2013. Quantum Databases. In *Sixth Biennial Conference on Innovative Data Systems Research, CIDR 2013, Asilomar, CA, USA, January 6-9, 2013, Online Proceedings.* www.cidrdb.org.

[46] Sartaj Sahni. 1975. Approximate Algorithms for the 0/1 Knapsack Problem. *J. ACM* 22, 1 (jan 1975), 115–124.

[47] Rainer Schlosser, Jan Kossmann, and Martin Boissier. 2019. Efficient scalable multi-attribute index selection using recursive strategies. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1238–1249.

[48] Karl Schnaitter, Serge Abiteboul, Tova Milo, and Neoklis Polyzotis. 2006. COLT: continuous on-line tuning. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data* (Chicago, IL, USA) *(SIGMOD '06)*. Association for Computing Machinery, New York, NY, USA, 793–795.

[49] 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.

[50] Transaction Processing Council. [n.d.]. *TPC-H*. www.tpc.org/tpch

[51] Immanuel Trummer and Christoph Koch. 2016. Multiple query optimization on the D-Wave 2X adiabatic quantum computer. *Proc. VLDB Endow.* 9, 9 (may 2016), 648–659.

[52] Gary Valentin, Michael Zuliani, Daniel C Zilio, Guy Lohman, and Alan Skelley. 2000. DB2 advisor: An optimizer smart enough to recommend its own indexes. In *Proceedings of 16th International Conference on Data Engineering (Cat. No. 00CB37073)*. IEEE, 101–110.

[53] Johannes Weidenfeller, Lucia C Valor, Julien Gacon, Caroline Tornow, Luciano Bello, Stefan Woerner, and Daniel J Egger. 2022. Scaling of the quantum approximate optimization algorithm on superconducting qubit based hardware. *Quantum* 6 (2022), 870.

[54] Sören Wilkening, Andreea-Iulia Lefterovici, Lennart Binkowski, Michael Perk, Sándor Fekete, and Tobias J. Osborne. 2023. A quantum algorithm for the solution of the 0-1 Knapsack problem. arXiv:2310.06623 https://arxiv.org/abs/2310.06623

[55] 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* (Seattle, WA, USA) *(BiDEDE '23)*. Association for Computing Machinery, New York, NY, USA, Article 5, 7 pages.

[56] Tobias Winker, Sven Groppe, Valter Uotila, Zhengtong Yan, Jiaheng Lu, Maja Franz, and Wolfgang Mauerer. 2023. Quantum Machine Learning: Foundation, New Techniques, and Opportunities for Database Research. In *Companion of the 2023 International Conference on Management of Data* (Seattle, WA, USA) *(SIGMOD '23)*. Association for Computing Machinery, New York, NY, USA, 45–52.

[57] Gongsheng Yuan, Jiaheng Lu, Yuxing Chen, Sai Wu, Chang Yao, Zhengtong Yan, Tuodu Li, and Gang Chen. 2023. Quantum Computing for Databases: A Short Survey and Vision. 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.

[58] Daniel C. Zilio, Jun Rao, Sam Lightstone, Guy Lohman, Adam Storm, Christian Garcia-Arellano, and Scott Fadden. 2004. DB2 design advisor: integrated automatic physical database design. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30* (Toronto, Canada) *(VLDB '04)*. 1087–1097.