A Divergent Index Tuning Advisor Using Quantum Machine Learning for Distributed Databases

Sam Bird
School of Computer Science
University of Oklahoma
Norman, OK, USA
sam.bird@ou.edu

Le Gruenwald
School of Computer Science
University of Oklahoma
Norman, OK, USA
ggruenwald@ou.edu

Sven Groppe
Institute of Computer Science
TU Bergakademie Freiberg, Freiberg, Germany, and
Institute of Information Systems
University of Lübeck, Lübeck, Germany
sven.groppe@uni-luebeck.de

***Abstract*—Automatic index selection is a critical task for optimising query performance in distributed database systems. This problem is NP-hard, and existing approaches based on heuristic methods or optimisation problems struggle to generate good recommendations within a reasonable timeframe for large-scale database applications. While using classical deep reinforcement learning with Deep Q-Networks (DQN) has shown promise, its scalability in high-dimensional state spaces remains a challenge. In this paper, we explore the application of hybrid quantum-classical deep reinforcement learning, specifically a Quantum Deep Q-Network (Q-DQN), to the divergent index tuning problem. Using the TPC-H benchmark on a varying number of database replicas, we compare the performance of classical divergent index tuning against a hybrid quantum-classical algorithm using a Q-DQN implemented on the IBM Qiskit Aer simulator. We find that the hybrid quantum-classical algorithm exhibits remarkably higher sample efficiency, converging in significantly fewer action steps while maintaining a comparable query execution time. Overall, our findings suggest that variational quantum circuits provide an effective representation for deep reinforcement learning, achieving faster policy convergence, and that when appropriately configured, this improved sample efficiency can be obtained without loss in query execution performance.**

***Index Terms*—index selection, replicated database, machine learning, quantum computing**

I. Introduction

An important aspect of administering a database is the selection of indexes to reduce query execution time, and attempts to automate their selection have been the subject of ongoing research [1]. Given a database of tables with attributes (columns) and tuples (records), the index selection problem attempts to minimise the cost to execute a workload of queries, given a space budget to create indexes [2]. This problem is generalised to that of *divergent design index selection*, where an algorithm aims to minimise the query workload processing cost over a cluster of fully replicated databases [3]. Here, an algorithm must generate both an index configuration for each database replica and a routing table that determines on which replica a given query in the workload will be processed.

Divergent design index tuning allows each replica in the distributed system to be specialised for a subset of the workload. This specialisation allows the system to process the overall workload faster compared to a uniform index configuration where each replica has the same indexes. The index selection problem is NP-hard [4], and the divergent design index selection problem is similarly difficult. Accordingly, several heuristic, optimisation, and machine learning approaches to the problem have been proposed [5]–[7].

Research on divergent index selection is more sparse. Consens et al. [3] first stated the problem, and Tran et al. [5] proposed RITA, another optimisation-based technique, to solve the problem. The state-of-the-art for divergent index tuning is given by Sadri et al. [6], who propose the Divergent Index Tuning Advisor (DINA), and an algorithm introduced by Hang et al. [7]. Both DINA and Hang's algorithm represent the latest approaches in divergent index tuning. However, our own experiments have shown that DINA outperforms Hang's algorithm by about 8% using the TPC-H queries-per-hour processed performance metric at scale factor 10 (QphH@10GB), so DINA remains the state-of-the-art in this space.

Attempts have been made to leverage a quantum advantage to improve recommendations for index selection. Gruenwald et al. [8] first proposed the use of quantum computing to tackle the index selection problem. Since then, various quantum and hybrid quantum-classical algorithms have been proposed. Barbosa et al. [9] propose QRLIT, which uses reinforcement learning to train an oracle to perform a Grover search over the set of index configurations to generate indexes. Trummer et al. [10] instead model index selection as an optimisation problem, then uses quantum annealing to find the optimal solution and recommend an index configuration. Kesarwani et al. [11] develop both Grover search and optimisation-based methods to solve index selection.

These approaches share two key characteristics: they target centralised database systems and frame index selection as a static optimisation problem. In contrast, divergent index tuning in replicated database clusters introduces a dynamic, learning-driven decision process, where index selection and query routing must adapt jointly to workload characteristics. This naturally motivates the use of quantum machine learning, rather than purely optimisation-based quantum techniques.

Quantum machine learning has shown promise in learning tasks such as classification, where some studies reach similar levels of accuracy with fewer training data [12], [13], but its

This work is supported in part by the National Science Foundation under grant No. 2425838.

applicability to adaptive database system problems such as divergent index tuning remains largely unexplored.

In this work, we evaluate whether quantum-enhanced reinforcement learning can be competitive in terms of convergence efficiency and query execution time for divergent index tuning on clusters of fully replicated databases under realistic workloads. We perform the first systematic study of quantum deep reinforcement learning for divergent index tuning, identifying when and why it differs from classical deep reinforcement learning and what this implies for future hybrid approaches. The remainder of this paper is structured as follows. Section II gives a background on quantum computing and discusses related work in quantum index tuning. Section III describes our proposed algorithm, the Quantum Divergent Index Tuning Advisor, qDINA. Section IV evaluates the performance of qDINA against the classical algorithm on TPC-H benchmark workloads and discusses its performance. Finally, Section V summarises our findings and presents a vision for future research in this area.

II. Related Work

A. Quantum computing

The basis of quantum information is the quantum bit, or qubit. Quantum information systems are more expressive than classical ones because a qubit can be in a superposition of the $|0\rangle$ state and the $|1\rangle$ state, while a classical bit is always in either one of those deterministic states. This superposition can be anywhere in a continuous probability distribution of being observed as one of the classical states. The state of a qubit is represented as $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers with a 2-norm of 1. Another means of representing the state of a qubit is as a point on the Bloch sphere, where qubit operations rotate the state around one or more of the three axes [14].

Qubits can be leveraged for computation by introducing entanglement into a system in superposition, where the probabilities of each qubit being observed as a given classical state become correlated. Entanglement enables correlations between qubits that underpin quantum algorithms offering asymptotic speedups over classical approaches [15].

Simulating a quantum computer is difficult because of the entanglement between the qubits. To encode the information in n qubits in a classical computer requires 2^n complex numbers. This difficulty leads to long computation times when attempting to simulate a quantum algorithm on a classical computer [16].

One approach for applying the representational power of quantum computers to existing problems is through quantum deep reinforcement learning. Unlike the classical method, quantum deep reinforcement learning uses a quantum neural network to approximate the reward function.

Quantum neural networks are typically implemented using a variational quantum circuit (VQC). A VQC applies a series of gates to one or more qubits in the quantum circuit, which affects their state [17]. For example, the R_z gate rotates the qubit around the z-axis of the Bloch sphere by some parameter. A VQC is parametrised by a set of input parameters, which encode the state of the environment, and a set of trainable parameters. Unlike classical neural networks, whose depth arises from stacking layers of neurons with affine transformations and nonlinear activation functions, the structure of a quantum neural network is determined by the layout of its circuit. Trainable parameters are introduced through rotation gates within the circuit's ansatz, which may consist of repeated blocks of parametrised rotations and entangling operations applied after input embedding. Analogous to weight updates in a classical neural network, adjusting these trainable parameters modifies the output probability distribution obtained upon measurement of the circuit.

This output can be used as input to further classical layers of a neural network, which are trained as any other classical network. Hybrid quantum-classical neural networks are useful for leveraging the benefits of quantum computing with limited resources in the noisy intermediate-scale quantum hardware (NISQ) era [18].

B. Quantum approaches to index tuning

The NP-hardness of the index selection problem has inspired attempts to leverage quantum computing to improve the speed and quality of recommendation algorithms [8]. Trummer et al. model index selection as an optimisation problem and apply a quantum algorithm to solve it [10]. Kesarwani et al. take a similar approach and apply a related quantum algorithm with OQIA [11]. They also present SQIA, which uses Grover's search algorithm directly for index selection. QRLIT [9] instead uses Grover search in a reinforcement learning context, training an oracle to return the action with the highest predicted Q-value within a learning episode.

Quantum machine learning holds several advantages over its classical counterpart. It is able to generalise better than standard techniques, particularly over smaller datasets [13]. Additionally, when considering deep learning, a parametrised quantum circuit can be more expressive than a classical neural network with a smaller number of trainable parameters [19]. Accordingly, we investigate the use of deep Q-learning with a parametrised quantum circuit in place of a neural network on the divergent index tuning problem, adapting DINA into a classical-quantum hybrid algorithm.

Our approach differs from existing algorithms in several important ways. Unlike Trummer's algorithm, SQIA, and OQIA, our algorithm does not require a priori knowledge of the benefits of constructing a given index, nor does it assume that there can be no overlapping benefit of each index. Instead, qDINA learns how each index is used in practice over the entire workload, whether that be from the query planner's cost estimation or the actual query execution times. Unlike QRLIT, qDINA is designed for divergent design index tuning (whereas QRLIT is a centralised algorithm).

III. Proposed Quantum Divergent Index Tuning Advisor (qDINA)

qDINA uses deep reinforcement learning with a quantum neural network to learn index configurations for each replica in the system. It consists of three modules: the workload parser, the learner, and the router. The parsing module reads in the query workload to extract index candidates, the learner attempts to find the best index configuration, and the router assigns each query template to the replica where it performs best. The output from the system is a set of index configurations for each replica and a routing table for each template in the workload.

A. Problem formulation

The objective is to minimise the execution cost of a workload of queries by creating indexes on each database replica, as well as a routing function that determines which replica a given query is sent to. Given replicas $r \in \{1, 2, ..., n\}$, a workload of queries $\mathcal{Q} = \{q_1, ..., q_k\}$, the template each query is drawn from $\mathcal{T} = \{qt_1, ..., qt_m\}$, index candidates $\mathcal{I}$, and an index space budget B, a divergent index tuning algorithm creates index configurations $I_r \subseteq \mathcal{I}$ for each replica and a routing function $\pi(qt) \to r$ that determines where a query (of a given template) is sent.

The goal is to minimise the processing time of the entire query workload. Let Q_r be the queries that are routed to the rth replica by the routing function π and let $cost_r(q)$ be the time it takes to execute query q on replica r. The goal is to minimise

$$\max_{r \in R} \sum_{q \in Q_r} cost_r(q)$$

Subject to the constraint that $|I_r| \leq B$ for all $r \in R$; that is, that the index space budgets are satisfied. This objective naturally induces long-term dependencies between indexing and routing decisions, making it well suited to reinforcement learning.

B. Reinforcement learning

qDINA uses a deep reinforcement learning algorithm to explore the space of possible index configurations and routing functions, evaluating their performance relative to the baseline case where no indexing is used. In reinforcement learning, an agent at timestep t has a current environmental state s_t, and will proceed to state s_{t+1} by taking some action a_t from the set of possible actions A. Which action the agent should take is chosen randomly during exploration, or is the action that has the maximum expected reward (Q value) during exploitation. The exploration probability ε decays over time, allowing the agent to rely more on prior inferences as training progresses.

Each learning step of the algorithm returns a (state, action, reward, next state) transition that is used to update the weights of a neural network that approximates the Q-function. qDINA uses a quantum neural network to perform reinforcement learning instead of a classical neural network or Q-table. The memory requirements needed to store expected reward values, particularly in big data applications, preclude storing the full Q-table under realistic hardware assumptions.

The reward is given by a combination of a *workload reward*, which increases as the cost of executing the query workload decreases due to the indexes present, and a *skew reward*, which decreases if the processing costs on the replicas become imbalanced. In this way, the learning agent is incentivised to choose good indexes that spread the processing cost evenly across the cluster of database replicas.

Finally, qDINA creates a routing function based on a 'best-fit' method. Each query belongs to a template, and the template is routed to the replica where its processing cost is lowest. The routing function is updated after each learning step to reflect the new index configurations present on each replica.

C. State representation

qDINA employs a compact, structured state representation designed to be compatible with parameterised quantum circuits. Each action that the reinforcement learning agent takes is the addition of an index to a database replica. The current state of the environment is represented as a state matrix, where each row represents a database replica and each column represents an index candidate. A 1 encodes the presence of an index candidate on a given replica, while a 0 encodes its absence.

However, while this sparse matrix is useful for the reinforcement learning agent, the use of a parameterised quantum circuit for an approximation of the Q-function requires the state input to be lower-dimensional. The input parameters to the quantum circuits are rotation values for each of the qubits in the $[0, 2\pi)$ range, which is incompatible with the native representation used by the reinforcement learning (RL) agent. Accordingly, qDINA encodes the state for the quantum circuit as a fixed-length real-valued vector.

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \mapsto [1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0]$$

$$\mapsto \left[(1\ \ 1\ \ 0)\ \ (0\ \ 0\ \ 0)\ \ (0\ \ 1\ \ 1)\ \ (0\ \ 1\ \ 0)\right] \mapsto [6\ \ 0\ \ 3\ \ 2] \mapsto \left[\tfrac{2\pi}{6}\ \ 0\ \ \tfrac{2\pi}{3}\ \ \tfrac{2\pi}{2}\right]$$

Fig. 1: Encoding process to prepare the state matrix for embedding into the input parameters of the quantum circuit, using 4 qubits as an example.

The full encoding process for the input to the VQC is given in Figure 1. First, the sparse state matrix is flattened into a vector. Then, the flattened matrix is divided amongst the qubits of the quantum circuit. The segments of the input vector are interpreted as binary integers. Finally, these integers are mapped to the input dimension of the circuit parameters through the transformation $x \mapsto 2\pi/x$ if $x \neq 0$, or $x \mapsto 0$ otherwise. This mapping preserves ordinal distinctions between index configurations while compressing large integer values into the bounded domain required by rotation gates. This representation differs from classical index tuning advisors, which typically operate on higher-dimensional sparse states.

D. Quantum neural network

In qDINA, a hybrid classical-quantum neural network is used to approximate the Q-function. This hybrid model uses the expressivity of the quantum neural network to capture the complexity of the Q-function in a smaller number of network parameters than a classical deep neural network, then uses a classical layer to map the output of the quantum neural network (QNN) to actions that the reinforcement learner can take.

As discussed in Section II-A, a QNN has three parts: *input parameters*, the *ansatz*, and the *output*. The input parameters are computed through the process discussed in Section III-C. To encode the input, a ZZ feature map is used. First, each qubit enters a superposition through the application of a Hadamard gate. Next, an R_z gate rotates the qubit position by the amount specified by each input parameter. Next, entangling controlled-NOT (CNOT) gates are interleaved with additional rotation gates which combine the input parameters for expressivity. This is relatively hardware-efficient while also introducing enough complexity to map the input to the higher-dimensional feature space of the quantum circuit [20], making it well suited for our problem of approximating a complex Q-function.

Next, the quantum neural network has an ansatz to introduce trainable parameters into the network. qDINA uses a hardware-efficient real amplitude ansatz, alternating trainable parameters (rotation gates around the Bloch sphere's Y-axis) and CNOT gates for further entanglement. The number of repetitions, and therefore the number of trainable parameters, impacts the expressive power of the QNN. Decreasing the number of repetitions causes the expressivity to suffer, while increasing the repetitions will lead to diminishing returns in noiseless simulation and qubit decoherence in real-world execution as noise is introduced through more gate operations. The best choice of ansatz is problem-dependent, but the general-purpose circuit we employ here is relatively hardware efficient (using only rotation and CNOT gates) and equally as expressive as alternatives under the number of repetitions that we study [21].

Measurement samples a bitstring from the circuit's output probability distribution in the computational basis. Each bitstring is mapped to an action for the agent to take. Because there is no way to know if a single output from the neural network is representative of the probability distribution of the circuit, several samples (shots) are taken. If a bitstring that corresponds to a given action is sampled more frequently from the network, it is interpreted as having a higher predicted Q-value relative to other actions.

Because the network can produce 2^Q bitstrings, where Q is the number of qubits, there are in practice usually more possible bitstrings than actions. Accordingly, qDINA uses a final classical neural network layer to map output bitstrings to actions. This allows the full expressiveness of the quantum circuit to be leveraged, as otherwise bitstrings that do not correspond to any action would have to be discarded.

E. Training and optimisation procedure

Training in qDINA follows an episodic reinforcement learning procedure. In each episode, the agent selects an update to the current index configuration, evaluates the resulting workload execution cost and skew, and computes a reward signal that reflects the improvement in performance. Parameter updates are performed at the end of each episode.

The trainable parameters of the quantum neural network, along with the weights of the classical output layer, are optimised using the simultaneous perturbation stochastic approximation (SPSA) algorithm [22]. SPSA does not require explicit gradient evaluations. Instead, it estimates the gradient by applying random perturbations to the parameters and resampling the objective function, significantly reducing the number of function evaluations required per update. This property is particularly advantageous in quantum settings, where each function evaluation corresponds to executing a quantum circuit, which is computationally and financially expensive. Moreover, the inherent randomness of quantum measurements introduces noise into objective evaluations, making gradient-based methods that rely on accurate gradient estimates difficult to apply in practice. Prior work by Pellow-Jarman et al. [23] demonstrates that SPSA is more robust than gradient-based optimisers in the presence of realistic noise, while exhibiting comparable performance under noiseless simulation. As a result, SPSA is well suited for training hybrid quantum–classical models and can reduce overall optimisation overhead compared to classical optimisers such as Adam [24].

F. Relationship to classical DINA

qDINA is related to the reinforcement learning architecture of DINA, which uses classical deep reinforcement learning [6]. In particular, the reward function and ε-greedy exploration strategy are the same, and the routing strategy used in qDINA is consistent with the classical version. However, qDINA uses a different state encoding for use in the variational quantum circuit, and the function approximator used in action value estimation is fundamentally different. These differences result in distinct exploration–exploitation dynamics, leading to substantial differences in the RL agent's convergence behaviour and action efficiency.

IV. Performance Evaluation

A. Environment

We ran our experiments on the Cloudlab cluster [25] using one virtual machine for the index tuner and an additional virtual machine for each database replica. Each replica used PostgreSQL 17 with default tuning parameters and HypoPG 1.4.1. When evaluating the workload reward, the RL agent issued `EXPLAIN` calls to the PostgreSQL query planner.

Each query workload was drawn from the same training set of queries generated using the TPC-H `qgen` utility. 50 queries were generated from each of the 22 query templates, for a total workload size of 1,100 queries. To evaluate the index recommendation quality, the same set of queries used for training were executed on the database cluster, according to the

learned routing function. Each replica was populated with data from the TPC-H `dbgen` utility. The quantum circuits were simulated using Qiskit's `AerSimulator` using its noiseless statevector simulation mode. Each circuit evaluation sampled 4,096 shots.

To benchmark the performance of each generated index configuration and routing table, replicas were created on individual, physical `c6620` nodes on Cloudlab.

B. Experimental setup

TABLE I: Dynamic parameters

Parameter	Default value	Range of values
Number of replicas	4	2, 3, 4, 5, 6
Ansatz repetitions	10	5, 10, 15, 20

To explore the possible advantages of quantum-enhanced reinforcement learning for divergent index tuning, we investigated qDINA's performance on the TPC-H workload against the existing DINA algorithm. Using the generated index configurations for each replica and the routing tables for each query template, we then created a database cluster and evaluated the query execution time of the workload. The workload was simulated to arrive simultaneously, with queries executing sequentially on each replica, but the subsets of the workload routed to different replicas executed in parallel (so no replicas were idle until they had finished processing their workload). We then measured the execution time of the query workload, as well as the number of actions taken by the reinforcement learning agent during each learning step. Each action requires invoking the database cost estimator; thus, the number of actions directly corresponds to the overhead imposed on the production system during tuning.

The reinforcement learning algorithms are stochastic in which actions they choose to randomly explore, so the same workload and parameters may yield different index recommendations and routing tables. Accordingly, every experimental run was repeated five times and the average of those runs is reported in this paper.

1) Number of database replicas: As more replicas are added to the cluster, the set of possible index configurations becomes larger. The optimal division of the query workload changes as more resources become available, and the reinforcement learning agent has a larger state space to explore. Accordingly, we investigate whether the hybrid quantum-classical learner is able to better generalise from existing information than a purely classical deep neural network.

2) Ansatz configuration: Choosing the best ansatz is important for maximising the performance of the quantum neural network. Our ansatz alternates parametrised rotation and entanglement gates. Using too few repetitions limits the number of trainable parameters and could make the quantum circuit insufficiently expressive for a good approximation of the reward function. Using too many repetitions, however, makes it more challenging for the optimiser to train the parameters, which could lead to overfitting. Additionally, adding additional layers exacerbates noise issues and barren plateaus [26].

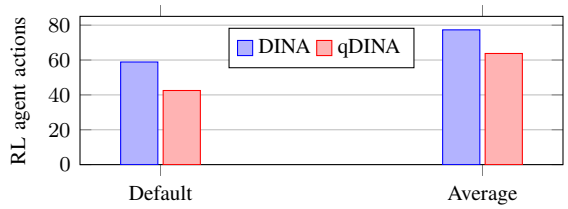

Fig. 2: Reinforcement learning agent actions taken per episode for the default configuration of 4 replicas and averaged over 2–6 replicas.

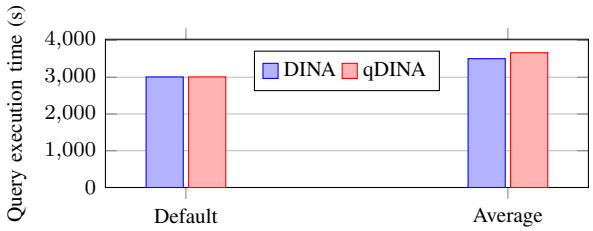

Fig. 3: TPC-H query workload execution time (1,100 queries) for the default configuration of 4 replicas and averaged over 2–6 replicas.

C. Experimental Results

1) Overall results: Across the evaluated configurations, qDINA converges more quickly than classical DINA, reducing the number of actions taken by the reinforcement learner by approximately 21% on average, while maintaining query execution times that are not statistically significantly different from the classical baseline. These results are shown in Figures 2 and 3, respectively.

Importantly, this trade-off is configuration-dependent. With four database replicas, qDINA matches the query execution time of classical DINA while requiring approximately 30% fewer actions, while with six replicas, qDINA achieves lower mean query execution time than DINA while requiring approximately 10% fewer actions. This indicates that, under appropriate system configurations, quantum-enhanced reinforcement learning can improve sample efficiency without sacrificing solution quality. This is particularly important in real-world database applications, where faster convergence allows automatic index tuning to be run more frequently and the continued execution of an algorithm that invokes the database cost estimator could degrade query response times.

Performing an analysis of variance test, we found that the sampling efficiency improvement in the number of RL agent actions taken was statistically significant ($p = 0.0163$), while the difference in query execution time was not ($p = 0.4748$).

These results are consistent with prior work on quantum deep RL, where quantum DQNs often converge to effective policies in fewer iterations, though final performance may depend on problem structure and parameterisation [27], [28].

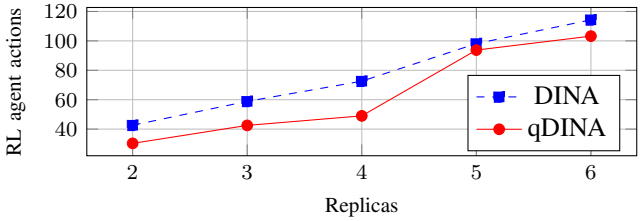

Fig. 4: Reinforcement learning agent actions taken per episode, by number of replicas.

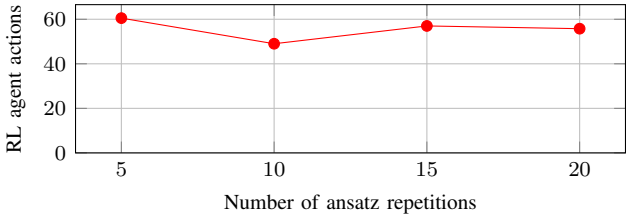

Fig. 6: Reinforcement learning agent actions taken per episode, by number of ansatz repetitions.

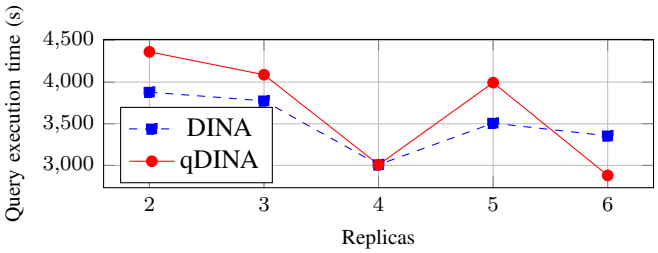

Fig. 5: TPC-H query workload execution time (1,100 queries), by number of replicas.

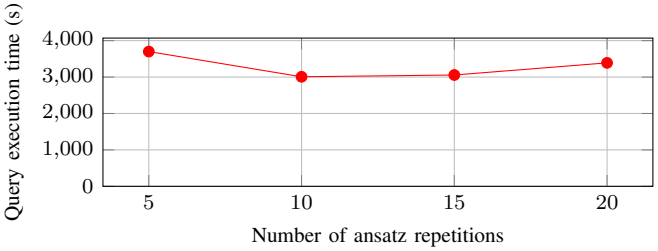

Fig. 7: TPC-H query workload execution time (1,100 queries), by number of ansatz repetitions.

2) Impact of dynamic parameters: Across a varying number of replicas, qDINA tended to show higher sample efficiency than DINA, as shown in Figure 4. This effect was most pronounced with 4 replicas, where the query execution times were nearly identical, as shown in Figure 5. However, the benefits decreased with 5 and 6 replicas in the cluster. Having held the number of qubits in the VQC constant at 8, the state encoding scheme discussed in Section III-C becomes more challenging for the quantum circuit to learn effectively. As the dimension of the state space increases, the size of the input parameter change that the quantum neural network must distinguish becomes smaller. The worse performance with more replicas suggests that 8 qubits is well matched to configurations with 4 replicas, but a larger state space requires more qubits. qDINA's encoding scheme allows it to use qubits more efficiently than competing quantum algorithms, which typically allocate one qubit for each index candidate, but also necessitates parameter tuning for the number of qubits on each number of replicas in the cluster.

Additionally, we observed that the sample efficiency gains are robust even at low circuit depths. As shown in Figures 6 and 7, there was not a substantial performance gain in either sample efficiency or query execution time achieved from increasing the number of ansatz repetitions from 5 to 20. This indicates that, for the TPC-H workload, a shallow quantum circuit is sufficient to capture the dominant correlations relevant to index selection. These results suggest that qDINA is able to achieve higher sample efficiency without the noise sensitivity typically associated with deeper quantum circuits.

V. Conclusion and Future Research

We presented qDINA, a hybrid quantum–classical divergent index tuning advisor that applies quantum deep reinforcement learning to the problem of index configuration and query routing in clusters of fully replicated databases. qDINA replaces the classical function approximator used in prior work with a parametrised quantum circuit, enabling more sample-efficient learning of indexing policies.

Across a range of experimental configurations, qDINA consistently requires fewer interactions with the database environment than classical DINA, reducing the number of actions taken by the reinforcement learner by an average of 21%, while achieving comparable query execution performance. Notably, for clusters with four replicas, qDINA converges 32% faster while producing index configurations with a statistically identical execution time to the classical baseline. These results indicate that quantum-enhanced reinforcement learning can meaningfully reduce tuning overhead without sacrificing solution quality when system and model parameters are appropriately aligned.

Our findings suggest that quantum deep reinforcement learning is a promising direction for automated index tuning in distributed database systems, particularly in settings where reducing tuning overhead is critical. Future work includes systematically studying the relationship between the size of the reinforcement learning state space and the number of qubits required, as well as evaluating qDINA on real quantum hardware.

Reproducibility

The source code for qDINA is available at https://github.com/const-sambird/dina. The source code for the query performance benchmarks is available at https://github.com/const-sambird/qdina-bench.

REFERENCES

[1] S. Agrawal, S. Chaudhuri, L. Kollar, A. Marathe, V. Narasayya, and M. Syamala, "Database tuning advisor for microsoft sql server 2005: demo," in *Proc. SIGMOD '05*, 2005, p. 930–932.

[2] D. Comer, "The difficulty of optimum index selection," *ACM Trans. Database Syst.*, vol. 3, no. 4, p. 440–445, Dec. 1978. [Online]. Available: https://doi.org/10.1145/320289.320296

[3] M. P. Consens, K. Ioannidou, J. LeFevre, and N. Polyzotis, "Divergent physical design tuning for replicated databases," in *Proc. SIGMOD '12*. New York, NY, USA: Association for Computing Machinery, 2012, p. 49–60. [Online]. Available: https://doi.org/10.1145/2213836.2213843

[4] S. Chaudhuri, M. Datar, and V. Narasayya, "Index selection for databases: a hardness study and a principled heuristic solution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1313–1323, 2004.

[5] Q. T. Tran, I. Jimenez, R. Wang, N. Polyzotis, and A. Ailamaki, "Rita: an index-tuning advisor for replicated databases," in *Proc. SSDBM '15*. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2791347.2791376

[6] Z. Sadri and L. Gruenwald, "A divergent index advisor using deep reinforcement learning," in *Proc. DEXA '22*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 139–152. [Online]. Available: https://doi.org/10.1007/978-3-031-12423-5_11

[7] H. Hang, X. Tang, B. Zhou, and J. Sun, "Unlocking the power of diversity in index tuning for cluster databases," in *Proc. DEXA '24*, C. Strauss, T. Amagasa, G. Manco, G. Kotsis, A. M. Tjoa, and I. Khalil, Eds. Cham: Springer Nature Switzerland, 2024, pp. 185–200.

[8] L. Gruenwald, T. Winker, U. Çalıkyılmaz, J. Groppe, and S. Groppe, "Index tuning with machine learning on quantum computers for large-scale database applications," in *Joint Workshops at 49th International Conference on Very Large Data Bases (VLDBW'23) — International Workshop on Quantum Data Science and Management (QDSM'23)*, 2023. [Online]. Available: https://ceur-ws.org/Vol-3462/QDSM5.pdf

[9] D. Barbosa, L. Gruenwald, L. D'Orazio, and J. Bernardino, "Qrlit: Quantum reinforcement learning for database index tuning," *Future Internet*, vol. 16, no. 12, 2024. [Online]. Available: https://www.mdpi.com/1999-5903/16/12/439

[10] I. Trummer and D. Venturelli, "Leveraging quantum computing for database index selection," in *Proc. Q-Data '24*. New York, NY, USA: Association for Computing Machinery, 2024, p. 14–26. [Online]. Available: https://doi.org/10.1145/3665225.3665445

[11] M. Kesarwani and J. R. Haritsa, "Index advisors on quantum platforms," *Proc. VLDB Endow.*, vol. 17, no. 11, p. 3615–3628, Jul. 2024. [Online]. Available: https://doi.org/10.14778/3681954.3682025

[12] D. Peral-García, J. Cruz-Benito, and F. J. García-Peñalvo, "Systematic literature review: Quantum machine learning and its applications," *Computer Science Review*, vol. 51, p. 100619, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013724000030

[13] M. C. Caro, H.-Y. Huang, M. Cerezo, K. Sharma, A. Sornborger, L. Cincio, and P. J. Coles, "Generalization in quantum machine learning from few training data," *Nature Communications*, vol. 13, no. 1, 2022. [Online]. Available: https://www.proquest.com/scholarly-journals/generalization-quantum-machine-learning-few/docview/2705220370/se-2

[14] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[15] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219. [Online]. Available: https://doi.org/10.1145/237814.237866

[16] A. Cicero, M. A. Maleki, M. W. Azhar, A. F. Kockum, and P. Trancoso, "Simulation of quantum computers: Review and acceleration opportunities," *ACM Transactions on Quantum Computing*, vol. 7, no. 1, Nov. 2025. [Online]. Available: https://doi.org/10.1145/3762672

[17] M. Cerezo, A. Arrasmitha, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, "Variational quantum algorithms," *Nat Rev Phys*, vol. 3, pp. 625 – 644, 2021.

[18] M. A. Hafeez, A. Munir, and H. Ullah, "H-qnn: A hybrid quantum–classical neural network for improved binary image classification," *AI*, vol. 5, no. 3, pp. 1462–1481, 2024. [Online]. Available: https://www.mdpi.com/2673-2688/5/3/70

[19] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, "Expressive power of parametrized quantum circuits," *Phys. Rev. Res.*, vol. 2, p. 033125, Jul 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.2.033125

[20] V. Havlicek, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, Mar. 2019.

[21] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1900070, 2019. [Online]. Available: https://advanced.onlinelibrary.wiley.com/doi/abs/10.1002/qute.201900070

[22] IBM, "SPSA – Qiskit Algorithms – IBM Quantum," 2025. [Online]. Available: https://qiskit-community.github.io/qiskit-algorithms/stubs/qiskit_algorithms.optimizers.SPSA.html

[23] A. Pellow-Jarman, S. McFarthing, I. Sinayskiy, D. K. Park, A. Pillay, and F. Petruccione, "The effect of classical optimizers and ansatz depth on qaoa performance in noisy devices," *Scientific Reports*, vol. 14, p. 16011, 2024.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[25] D. Duplyakin, R. Ricci, A. Maricq, G. Wong, J. Duerig, E. Eide, L. Stoller, M. Hibler, D. Johnson, K. Webb, A. Akella, K. Wang, G. Ricart, L. Landweber, C. Elliott, M. Zink, E. Cecchet, S. Kar, and P. Mishra, "The design and operation of CloudLab," in *Proceedings of the USENIX Annual Technical Conference (ATC)*, Jul. 2019, pp. 1–14. [Online]. Available: https://www.flux.utah.edu/paper/duplyakin-atc19

[26] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, "Cost function dependent barren plateaus in shallow parametrized quantum circuits," *Nature Communications*, vol. 12, no. 1, p. 1791, Mar. 2021.

[27] H.-Y. Chen, Y.-J. Chang, S.-W. Liao, and C.-R. Chang, "Deep q-learning with hybrid quantum neural network on solving maze problems," *Quantum Machine Intelligence*, vol. 6, no. 1, p. 2, 2024.

[28] H. Hohenfeld, D. Heimann, F. Wiebe, and F. Kirchner, "Quantum deep reinforcement learning for robot navigation tasks," *IEEE Access*, vol. 12, p. 87217–87236, 2024. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2024.3417808