Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue

Cite this: *Digital Discovery*, 2025, **4**, 2229

Generative quantum combinatorial optimization by means of a novel conditional generative quantum eigensolver†

Shunya Minami, *[a] Kouhei Nakaji,[bc] Yohichi Suzuki,[a] Alán Aspuru-Guzik [cd] and Tadashi Kadowaki[ae]

Quantum computing is entering a transformative phase with the emergence of logical quantum processors, which hold the potential to tackle complex problems beyond classical capabilities. While significant progress has been made, applying quantum algorithms to real-world problems remains challenging. Hybrid quantum-classical techniques have been explored to bridge this gap, but they often face limitations in expressiveness, trainability, or scalability. In this work, we introduce conditional Generative Quantum Eigensolver (conditional-GQE), a context-aware quantum circuit generator powered by an encoder–decoder transformer. Focusing on combinatorial optimization, we train our generator for solving problems with up to 10 qubits, exhibiting nearly perfect performance on new problems. By leveraging the high expressiveness and flexibility of classical generative models, along with an efficient preference-based training scheme, conditional-GQE provides a generalizable and scalable framework for quantum circuit generation. Our approach advances hybrid quantum-classical computing and contributes to accelerate the transition toward fault-tolerant quantum computing.

Received 4th April 2025
Accepted 11th July 2025

DOI: 10.1039/d5dd00138b

rsc.li/digitaldiscovery

1 Introduction

We are at the dawn of the era of fault-tolerant quantum computation. Logical qubits have been demonstrated across several quantum computing architectures.[1–6] Recent advances in error-correcting codes[7,8] further accelerate the shift toward early fault-tolerant systems in the relatively near future. While these developments enable substantially more quantum operations than systems without error correction, executing fault-tolerant quantum algorithms for practically significant problems remains a distant goal. Thus, building hybrid quantum-classical algorithms that work with quantum devices in the early-fault-tolerant regime is a practical and reasonable focus[9,10] for near-term quantum computing applications.

One widely studied methodology over the past decade is the variational quantum algorithm (VQA).[11–14] Applications of VQA, such as quantum machine learning,[15–18] often require uploading classical data into the circuit. The most common strategy is to embed the data into the rotation angles of gates in a parameterized quantum circuit (PQC). However, this approach faces limitations in expressibility, as the Fourier components of the expectation function are constrained to specific wave numbers.[19–21] Moreover, embedding classical knowledge or inductive bias into the PQC structure remains challenging,[22] despite the critical role of inductive bias in successful optimization.[23] Addressing these limitations requires innovative strategies that lead to the next-generation hybrid quantum-classical computation.

This paper explores an alternative approach based on the recently proposed generative quantum eigensolver (GQE).[24] GQE is a hybrid quantum-classical algorithm that uses a classical generative model to construct quantum circuits, where circuit components are sequentially generated from a predefined gate pool, similar to sentence generation in natural language processing. Unlike VQAs, no parameters are embedded in the quantum circuit; all parameters are contained within a classical generative model (see Fig. 1). These parameters are iteratively updated to minimize a particular objective. In a proof-of-concept experiment,[24] the generative model is implemented using GPT-2 (ref. 25) architecture, referred to as the generative pre-trained transformer quantum eigensolver (GPT-QE), and its effectiveness is demonstrated in the ground state search of molecular Hamiltonians. A key feature of GQE is its ability to incorporate classical variables directly into the neural network, allowing for a non-trivial influence on the generated quantum circuits. Additionally, inductive biases can

[a]*Global R&D Center for Business by Quantum-AI Technology, National Institute of Advanced Industrial Science and Technology, Ibaraki, Japan. E-mail: s-minami@aist.go.jp*

[b]*NVIDIA Corporation, 2788 San Tomas Expressway, Santa Clara, CA 95051, USA*

[c]*Chemical Physics Theory Group, Department of Chemistry, University of Toronto, Toronto, Ontario, Canada*

[d]*Vector Institute for Artificial Intelligence, Toronto, Ontario, Canada*

[e]*DENSO Corporation, Tokyo, Japan*

† Electronic supplementary information (ESI) available. See DOI: https://doi.org/10.1039/d5dd00138b

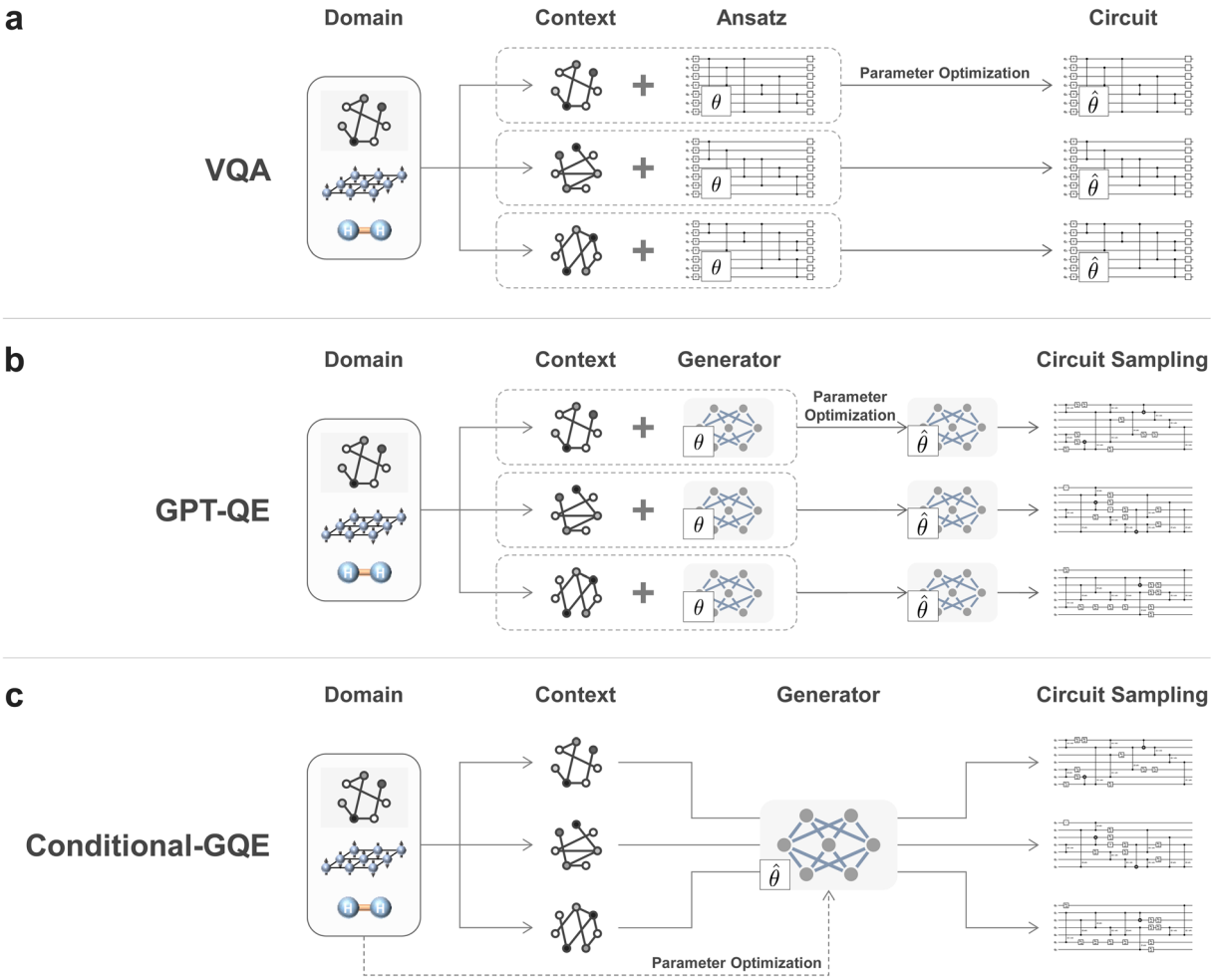

Fig. 1 Schematic of differences between VQA, GPT-QE, and conditional-GQE. (a) VQAs such as VQE prepare a parameterized quantum circuit, called ansatz, for each context (*i.e.*, target problem) and optimizes the parameters to minimize the expected value of the observables. (b) GPT-QE optimizes the parameters for each context; however, the parameters are given as weights in a classical neural network instead of being embedded in the quantum circuit. The final results are obtained by sampling circuits from the trained model. In the current version of the algorithm, one needs to retrain the model whenever a new problem is given. (c) This study develops a context-aware quantum circuit generator by using an encoder–decoder structure that enables the model to be conditioned on the problem context. Once trained, the model can be used for any context in the domain and does not necessarily need to be re-trained.

be seamlessly integrated, much like classical convolutional neural networks in computer vision[26–28] and graph neural networks in materials informatics.[29–31] While the potential of incorporating classical variables into the generative model has been previously discussed in the context of quantum chemistry,[24] specific methods for its implementation have not yet been detailed.

Based on the concept of GQE, this paper introduces conditional-GQE (Fig. 1c), an input-dependent quantum circuit generation. To generate circuits from given inputs, we adopt an encoder–decoder transformer architecture,[32] making the model applicable across different contexts. We apply this conditional-GQE approach to combinatorial optimization and develop a new hybrid quantum-classical method called Generative Quantum Combinatorial Optimization (GQCO). By incorporating a graph neural network[33] into the encoder to capture the underlying graph structure of combinatorial optimization problems, our model is trained to generate quantum circuits to solve combinatorial optimization problems with up to 10 qubits, achieving about 99% accuracy on new test problems. Notably, for 10-qubit problems, the trained model finds the correct solution faster than brute-force methods and the quantum approximate optimization algorithm (QAOA).[34]

Many of the existing works for quantum circuit design[35,36] often rely on labeled datasets, which limits their scalability as classical simulation becomes infeasible for large circuits. Although some recent approaches explore reinforcement learning for circuit optimization,[37,38] they typically require computing intermediate quantum states to guide gate selection. Consequently, both these supervised and reinforcement learning methods become impractical for large-scale quantum systems where classical simulation of the quantum algorithm is not feasible. To address these challenges, we introduce a dataset-free, preference-based algorithm. Specifically, this work

uses direct preference optimization (DPO)[39] to update the circuit parameters by comparing the expected values of generated circuits. Unlike many supervised or reinforcement learning-based methods, our DPO-based strategy does not rely on prior labeling; it only requires the final measurement results of the generated circuits, thereby substantially reducing computational overhead.

As an illustrative demonstration of conditional-GQE, we focus on combinatorial optimization problems. However, the goal of this study is not to outperform existing state-of-the-art methods in combinatorial optimization. Indeed, a wide range of solvers have been developed, including traditional algorithms like simulated annealing (SA),[40] machine learning-based approaches,[41,42] quantum annealing,[43] and techniques based on classical and quantum generative models.[44] Rather than aiming to surpass these existing methods, our broader goal is to present a novel, scalable, and generalizable workflow for quantum circuit generation across diverse domains, which is accelerated with the help of high-performance computing systems.[10] This work is expected to support practical quantum computation in the early fault-tolerant era and advance quantum technology's real-world application.

2 Results

2.1 Generative quantum eigensolver (GQE)

Large language models (LLMs) generate sequences of token indices, each corresponding to a word or subword, which, in turn, form a sentence together. Analogously, GQE generates quantum circuits by mapping each index to a component of a quantum circuit, such as a gate or a gate combination. The generated sequence of indices results in a composition of quantum gates, forming a quantum circuit.

Given a fixed initial state $|\phi_{\text{ini}}\rangle$, GQE uses classical machine learning to generate a quantum circuit U that minimizes the expectation value $\langle \mathcal{O} \rangle_U := \langle \phi_{\text{ini}}|U^\dagger \mathcal{O} U|\phi_{\text{ini}}\rangle$ of an observable $\mathcal{O}$. In many quantum computing applications, observables can be expressed as the function $\mathcal{O}(x)$ of certain variables x, such as coefficients of the Ising Hamiltonian representing combinatorial optimization problems. However, similar to many VQAs, GPT-QE—the original demonstration of GQE—does not incorporate x into the generative model but instead uses a separate model for each context, as illustrated in Fig. 1a and b. We believe that incorporating contextual inputs into the generative model can yield significantly different results compared to previous algorithms. This study presents the context-aware GQE, which aims to train a generative model with contextual inputs, generating a circuit that minimizes the energy $\langle \mathcal{O}(x) \rangle_U$ in response to a given input x. In contrast to GPT-QE, which utilizes a decoder-only transformer, we employ a transformer architecture that includes both an encoder and a decoder. The details of GQE and our approach are provided in the Methods section.

In previous work by some of us,[45] we suggested a way of training a parameterized quantum circuit $U(\theta, x)$ depending on the variables x. In this algorithm, the variables x are embedded into the circuit, and the parameters θ are optimized so that $\langle \mathcal{O} \rangle_{U(\theta,x)}$ is minimized for each x. However, embedding classical data into a parameterized quantum circuit faces the challenge of expressibility,[19–21] meaning that the functional form of $U(\theta, x)$ for x is severely restricted. In contrast, in GQE, we are not restricted by these expressibility issues. The variables x are incorporated into the classical neural network alongside trainable parameters, and they affect non-trivially the generated quantum circuit.

2.2 Conditional quantum circuit generation for combinatorial optimization

As a very important practical application, we focus on solving combinatorial optimization problems with conditional-GQE, which we call Generative Quantum Combinatorial Optimization (GQCO). The schematic diagram of the entire workflow is shown in Fig. 2.

Combinatorial optimization problems can always be mapped to a corresponding Ising Hamiltonian,[46] which serves as an observable. We use the Hamiltonian coefficients as input to a generator, embedding them into graph structures with feature vectors that capture domain-specific information. A graph encoder with transformer convolution[47] then produces an encoded representation. Using this, a transformer decoder generates a sequence of token indices that defines a quantum circuit. The solution is identified by selecting the bit sequence corresponding to the computational basis state with the highest observation probability from the generated quantum circuit. Fig. 2 presents the schematic of this process, with further details provided in the Methods section.

Circuit component pools must be predefined, allowing for the incorporation of domain knowledge and inductive bias. For example, since GPT-QE[24] aims to search a ground state for a given molecule, the operator pool is composed of unitary coupled-cluster singles and doubles (UCCSD) ansatz[48] derived from the target molecule. In this study, we use basic 1- and 2-qubit gates (Hadamard gate, rotation gates, and CNOT gate) and the QAOA-inspired R_{ZZ} rotational gate, *i.e.*, an Ising-*ZZ* coupling gate acting on two target qubits. The target qubit(s) of each quantum gate and, if necessary, the control qubit, are available in all configurations, and there are six possible rotation angles of $\{\pm\frac{\pi}{3}, \pm\frac{\pi}{4}, \pm\frac{\pi}{5}\}$ for the rotation gates. By using basic gates rather than components suitable for many-body physics such as the UCCSD ansatz, this work aims to study whether we can train the model successfully without prior knowledge of an optimal or intuitively useful operator pool.

While a detailed description of our training strategy is provided in the Methods section, we summarize it here to highlight our scalable, broadly applicable framework. Scaling circuit size is critical for fault-tolerant quantum computing; however, most prior works[35,36] rely on supervised learning methods that struggle to produce high-quality training data at large scales. In contrast, GPT-QE employs an alternative training approach called logit matching. This method does not require any pre-existing dataset; instead, it trains the generative model to approximate a Boltzmann distribution derived from the expectation value of a given Hamiltonian. In this work, to

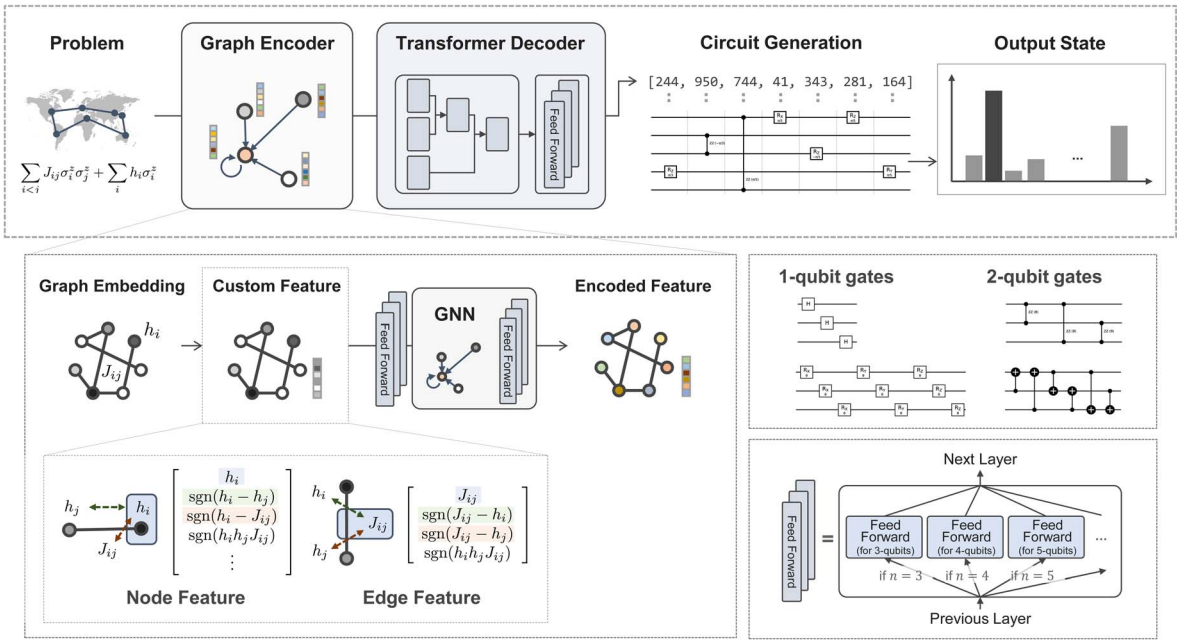

Fig. 2 Overview of generative quantum combinatorial optimization (GQCO). GQCO employs an encoder–decoder transformer architecture. The target combinatorial optimization problem is represented as a graph derived from the coefficients of the corresponding Ising model. Features are engineered based on domain knowledge, and an encoded feature representation is obtained using a graph neural network. The encoded feature is passed to a decoder transformer, which sequentially generates token indices and constructs sequences of 1- or 2-qubit quantum gates. The mixture-of-experts (MoE) architecture is used in the model structure to improve the model expressiveness. The solution to the input problem is obtained from the quantum states computed by the generated circuit.

further increase the probability around the preferred circuits beyond what is computed by the Boltzmann distribution, we use a preference-based strategy called direct preference optimization (DPO).[39] DPO compares candidate circuits based on their computed costs and updates the model parameters to increase the likelihood of the most favourable circuit. Crucially, it relies solely on expectation values from the generated circuits, eliminating the need for labelled datasets and therefore it facilitates the treatment of large-scale quantum systems. In other words, the model is trained by exploring the space of solutions rather than relying on previously-gathered ground truth. To manage the diversity arising from different problem sizes, we introduce a qubit-based mixture-of-experts (MoE) architecture.[49–51] This module comprises specialized model sublayers called experts, and the model switches between layers depending on the number of qubits required. We further accelerate model training through curriculum learning,[52] starting with smaller circuits and increasing task complexity step by step, then we proceed to fine-tune each expert for the respective problem size. Our preference-based curriculum training with MoE modules enhances the model's expressive power and scalability, facilitating the efficient integration of larger quantum circuits.

2.3 Solving combinatorial optimization *via* GQCO

We trained a GQCO model capable of generating quantum circuits with 3 to 10 qubits. All computations during the training were performed on classical hardware (CPUs and GPUs), and quantum calculations were conducted using a classical simulator. Specifically, multiple NVIDIA V100 GPUs were used for the GPU computations. Details of the training and hardware are provided in the Method section and the ESI.†

Fig. 3a compares the accuracy of GQCO with two other solvers—simulated annealing (SA)[40] and the quantum approximate optimization algorithm (QAOA)[34]—on 1000 randomly generated combinatorial optimization problems for each problem size. Both training and test datasets were generated from the same distribution. For each test problem, the GQCO model generated 100 circuits (the same number as the number used during training), and the circuit yielding the lowest expected value was selected. SA and QAOA were initialized and performed independently for each problem; in particular, in QAOA, the circuit parameters were trained from scratch for each problem and the solution was determined based on the optimized circuit. SA was executed with 1000 sweeps and 100 reads per problem instance, while QAOA utilized circuits with four layers. Results for the other hyperparameter settings are provided in Fig. 3c, and detailed descriptions of the configurations are provided in the Methods section.

As shown in Fig. 3a, the GQCO model consistently achieved a high accuracy of approximately 99% across all problem sizes. In contrast, QAOA failed to exceed 90% accuracy even for a 3-qubit task, and its accuracy declined to about 30% for a 10-qubit task. This performance drop reflects the limited expressive power and trainability of the canonical QAOA approach. Achieving over 90% accuracy with QAOA would require a much deeper parameterized circuit, making, at the current time,

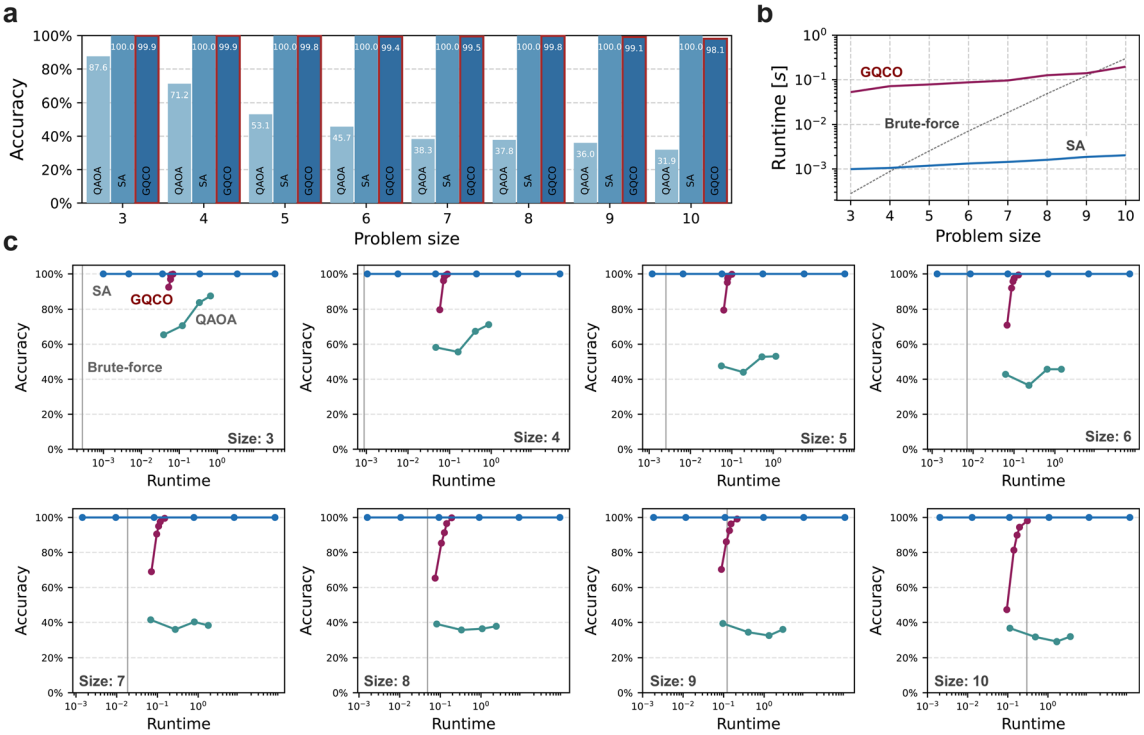

Fig. 3 Performance evaluation of GQCO and two other solvers. (a) Percentage of correct answers of QAOA, SA, and GQCO on 1000 randomly generated combinatorial optimization problems (3–10 qubits). (b) Runtime required to reach 90% accuracy. The red line represents GQCO, the blue line represents SA, and the gray dashed line represents the brute-force calculations. QAOA is excluded as it did not achieve 90% accuracy. (c) Runtime *versus* accuracy across problem sizes. As in (b), the red lines correspond to GQCO, the blue lines to SA, and the green lines to QAOA. Gray vertical lines show brute-force execution times; points to the left indicate a faster runtime than brute force. The points for each solver correspond to varying parameter settings: the number of sampling circuits {1, 5, 10, 20, 100} for GQCO, the number of sweeps {102, 103, 104, 105, 106, 107} for SA, and the number of layers {1, 2, 3, 4} for QAOA.

stable training infeasible. In contrast, GQCO addresses these limitations by leveraging the high expressive power of classical neural networks and by employing a large number of parameters on the classical side of the computation. The performance gap observed here indicates the advantages of the generative quantum algorithm approach over variational algorithms.

Fig. 3b shows the time required for each method to reach 90% accuracy. To adjust runtime, we varied the hyperparameters—namely, the number of sampled circuits for GQCO, the number of sweeps for SA, and the number of iteration layers for QAOA. The total runtime includes all steps, from submitting a test problem to identifying the answer. For GQCO, this runtime encompasses both model inference and quantum simulation; for QAOA, it includes parameter optimization and quantum simulation. SA and brute-force calculations were performed on CPUs, while the other computations, including quantum simulation, were conducted on GPUs. The gray dashed line indicates the runtime of brute-force search, which grows exponentially with problem sizeIn contrast, the increase in GQCO's runtime was restrained. Although it took a certain amount of computation even for small problem sizes due to the need for transformer inference, GQCO surpassed the brute-force method when the problem size exceeded 10 qubits. In terms of computational complexity, the brute-force method for a problem size n requires a runtime on the order of $O(2^n)$. In contrast, GQCO's complexity depends on both transformer inference and the quantum computation of the generated circuit. The former depends on sequence length[53] (*i.e.*, circuit depth) and scales on the order of $O(n^2)$, while the latter can potentially benefit from exponential speedup on quantum devices. In other words, GQCO can be expected to provide polynomial acceleration compared to brute-force, though GQCO does not guarantee to reach 100% accuracy. It is important to note that, in this performance evaluation, the quantum computations were performed using a GPU-based simulator, so any speedup that could be gained from a quantum approach would not be present in any of these results. Nevertheless, a clear reduction in the growth rate of the runtime even for these classical simulations observed.

Fig. 3c illustrates the detailed relationship between runtime and accuracy when varying the hyperparameters for each method: the number of generated circuits for GQCO, the number of sweeps for SA, and the number of layers for QAOA. Generally, performance improved as execution time increased. However, for QAOA, increasing the number of layers did not consistently enhance the performance of the algorithm, especially with an increasing number of qubits. This behaviour is attributed to the training difficulties inherent in VQAs. In contrast, GQCO outperformed the other solvers, demonstrating greater performance gains as the execution time grew. This

advantage arises from the processing power of GPUs, which enables additional samplings at little additional wall-clock cost, thereby boosting performance.

Because the runtime baseline depends on a device's computational power, the problem size at which the advantage emerges may differ across devices. However, the difference in computational complexity is independent of the device used.

2.4 Performance under distribution shift

In the experiments shown in Fig. 3, both the training and test datasets were created by randomly sampling Hamiltonian coefficients from the same uniform distribution. While the GQCO model achieves high accuracy under these conditions, real-world applications typically involve Hamiltonians with more structured characteristics. To investigate performance under such realistic conditions, Fig. 4 evaluates the pretrained GQCO model on sparse, 10-node graph Max-Cut problems with varying connectivity levels. Note that the model evaluated in Fig. 4 is identical to that used in Fig. 3, which was trained on a non-structured dataset. Although the model generally maintains high accuracy, its performance declines when encountering highly structured problems. Specifically, for Max-Cut problems on 3-regular graphs, accuracy drop to approximately 95%. This decline likely occurs because randomly sampled training data rarely include sparse graphs with many zero coefficients, resulting in insufficient training examples for such specialized cases.

Ideally, randomly generated Hamiltonians should cover the entire solution space for combinatorial optimization problems involving a given number of qubits. In practice, however, limited training iterations prevent complete coverage of the problem space, restricting the model's performance on structured problems. This limitation can be addressed by fine-tuning the pretrained GQCO model specifically for targeted structured problems. As illustrated by the green bar in Fig. 4, fine-tuning the pretrained GQCO model on 3-regular graph Max-Cut problems successfully improves accuracy to about 98%, nearly matching performance levels observed for dense graphs. This result highlights the importance of fine-tuning for practical applications with structured optimization tasks.

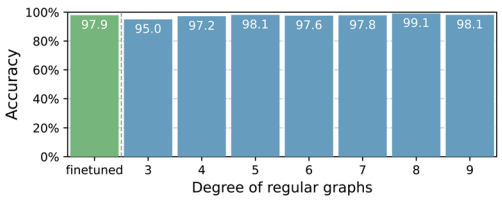

Fig. 4 Performance comparison of pretrained GQCO models on Max-Cut problems for sparse graphs with 10 nodes. Graph connectivity ranges from degree 3 to degree 9 (fully connected). Blue bars indicate the performance of the pretrained GQCO model, while the green bar represents performance after fine-tuning on the Max-Cut problem for 3-regular graphs. The fine-tuning was conducted for 1400 epochs with learning rate 1×10^{-7}.

2.5 Error analysis of GQCO solutions

During the performance evaluation, we identified one incorrect answer in each size-3 and size-4 problem set and two in the size-5 problem set. Fig. 5 shows the corresponding coefficient matrices, generated quantum circuits, and resulting quantum states. As mentioned above, we sampled 100 circuits for each problem, with the circuit yielding the lowest expected value among them identified as the GQCO answer. Note that lower-cost configurations may exist outside this finite sampling; indeed, for the four problems in Fig. 5, sampling 100 circuits alone did not produce a correct solution. In each case, the second-best solution had a cost that was very close to the optimal value, causing the GQCO model to become trapped in a near-suboptimal solution. This likely occurred because the transformer cannot fully capture the discrete nature of combinatorial optimization, where slight fluctuations in the Hamiltonian coefficients can lead to discontinuous changes in the solution. Increasing training time or using more precise floating-point calculations may help in reducing such errors. Another effective approach is to increase the number of circuit samplings; indeed, the four problems in Fig. 5a were correctly solved by GQCO when 1600, 300, 400, and 700 circuits were sampled, respectively. In natural language processing, the inference scaling law[54–56] states that increasing the inference time improves the quality of model outputs. A similar phenomenon appears to apply to quantum circuit generation as well. However, because generative models are inherently stochastic, theoretically guaranteeing perfect accuracy for circuit generation remains challenging.

2.6 Characteristics of generated circuits and limitations of GQCO

Examining the structure of circuits generated by GQCO offers insight into how GQCO solves problems. Fig. 6a and b show the average circuit depth and the number of CNOT gates for circuits generated by the GQCO model and a single-layer QAOA circuit, respectively. Both values were obtained after transpiling the circuits using Qiskit[57] with optimization level 1 (light optimization). Notably, the GQCO-generated circuits are shallower and include fewer CNOT gates than those produced by QAOA. Because the GQCO algorithm does not use the Hamiltonian directly in its circuit design, it does not require the extensive entanglement operations that QAOA does.

In this work, we did not impose an explicit restriction on the number of CNOT gates, although the maximum circuit depth for GQCO was set to twice the number of qubits. Certainly, the cost function and model structure are flexible enough to incorporate additional constraints on circuit depth or CNOT gate count. Further constraints that lead to shallower circuits could help generate circuits more robust to noise. Furthermore, the model can address device-related constraints. Because many quantum devices have restricted physical connectivity,[58,59] compilation is often needed to map circuits onto the hardware. However, GQE-based quantum circuit generators can bypass this process by excluding gates that do not satisfy the device's

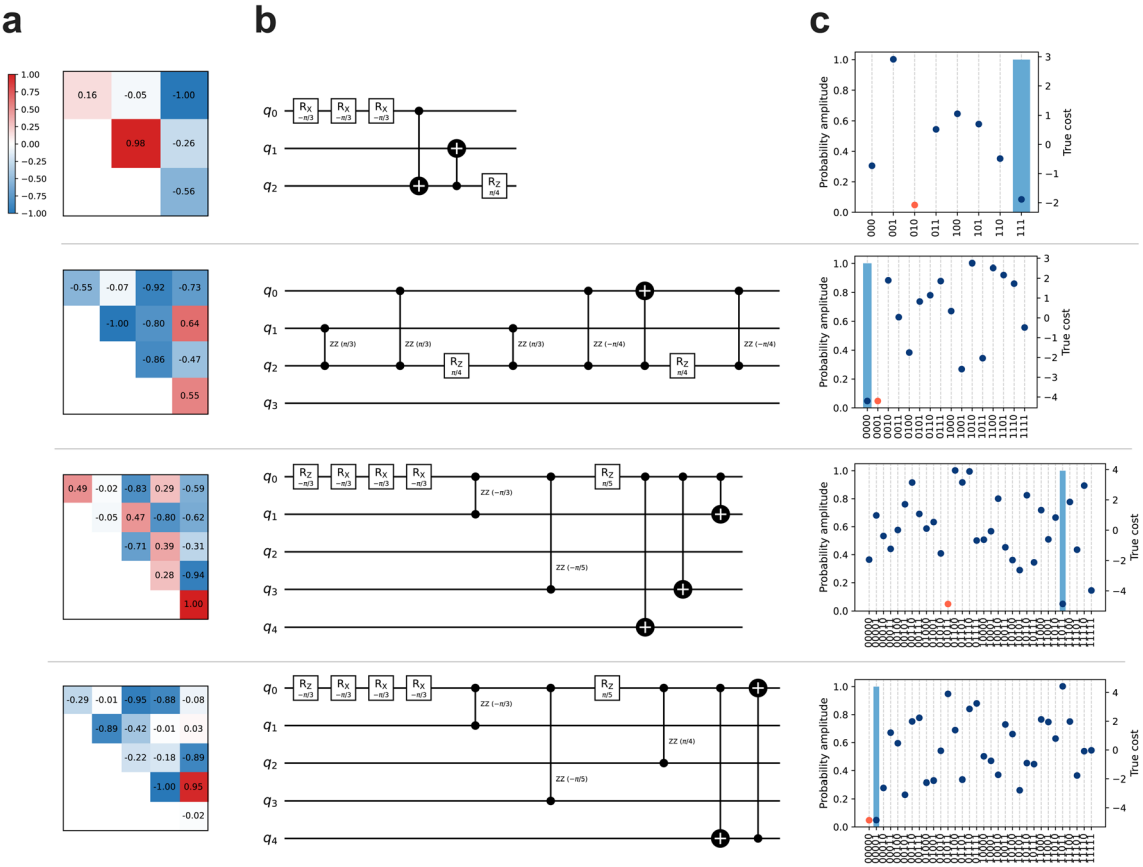

Fig. 5 Cases where the GQCO model failed to identify the correct solution. (a) Heat maps of the Ising Hamiltonian coefficient matrices for four incorrectly solved problems, with diagonal elements representing external fields and off-diagonal elements representing interaction terms. (b) Quantum circuits with the lowest expected energy, selected from 100 circuits generated by the GQCO model for each combinatorial optimization problem. (c) Corresponding quantum states obtained from these circuits. The histograms show observation probability for each computational basis (left y-axis) computed by state vector simulations, while the point plots show the Hamiltonian expectation value (*i.e.*, the cost of the combinatorial optimization problem) computed in each computational basis (right y-axis). The red dot for each plot corresponds to the basis with the lowest expected value, indicating the ground truth.

physical constraints. This flexibility in generating hardware-efficient circuits is a key advantage of the GQE-based approach.

Fig. 6 shows a typical 3-qubit quantum circuit generated by our GQCO model. In this circuit, six gates are used to transform the initial state $|000\rangle$ to state $e^{-i\frac{\pi}{10}}|001\rangle$. Notably, the three successive $R_Y(\pi/3)$ gates placed in the middle of the circuit are primarily responsible for obtaining the final state. The matrix representation of the composition of three $R_Y(\pi/3)$ gates is given by $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$, which corresponds to a bit flip from $|0\rangle$ to $|1\rangle$ (or

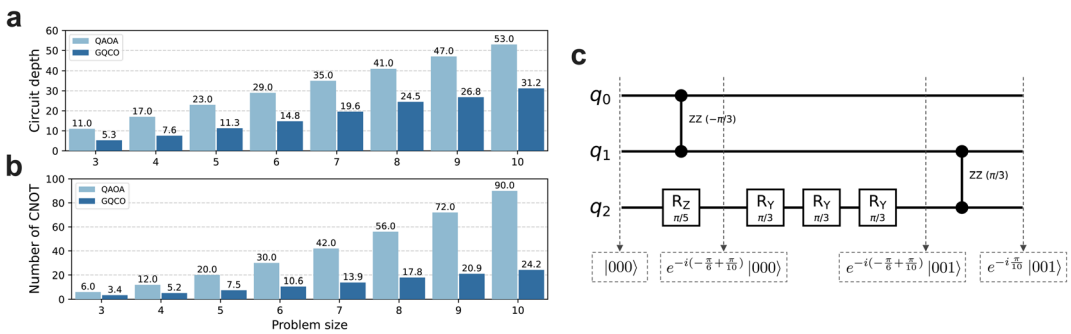

Fig. 6 Analyses of the generated circuits. (a) Comparison of circuit depth and (b) number of CNOT gates between GQCO-generated and one-layer QAOA circuits for each problem size. (c) A representative example of a generated circuit. Six quantum gates are applied sequentially to the initial state $|000\rangle$ to obtain the final quantum state $e^{-i\frac{\pi}{10}}|001\rangle$. The quantum states at two intermediate stpes are also illustrated.

from $|1\rangle$ to $-|0\rangle$). The remaining three gates (one R_Z gate and two R_{ZZ} gates) only change the global phase for the computational basis states and have no direct effect on the final solution. These observations suggest that GQCO differs substantially from quantum-oriented methods such as QAOA in that the GQCO model does not acquire a quantum mechanics-based logical reasoning capability. Instead, much like many classical machine learning models, GQCO appears to generate circuits by interpolating memorized instances. GQCO's circuit-generation ability relies on a data-driven approach rather than any logical understanding of quantum algorithms.

All of the circuits generated during the performance comparison (Fig. 3) are non-Clifford circuits and are generally expected to be difficult to simulate classically. However, as noted above, many of these circuits primarily perform bit flips. If we remove gates that affect only the global phase (*e.g.*, the first R_{ZZ} gate in Fig. 6c), most GQCO-generated circuits become Clifford, allowing them to be classically simulable. Consequently, our findings do not demonstrate a quantum advantage[58] or the quantum utility[59] of GQE-based circuit generation. Nevertheless, even if the trained model produces circuits that can be classically simulated, non-Clifford circuits are still generated during training. In other words, the entire circuit space—including circuits that are computationally hard to simulate classically—must be explored to obtain the trained model, highlighting the benefits of incorporating quantum computation into the overall workflow. Moreover, for applications beyond combinatorial optimization, solutions often involve more complex quantum states, and the circuits generated by the trained model are expected to be classically unsimulable. Since our model can be trained without explicitly determining whether the generated circuits are classically simulable, the GQCO workflow applies equally well to problems that rely on superposition or entanglement.

These results do not imply that GQCO is inherently a classical heuristic method. Indeed, the original GQE[24] has demonstrated strong performance on quantum tasks, such as the ground-state searches for molecules. In this study, the absence of quantumness in the generated circuits is likely because the combinatorial optimization problem itself is not intrinsically quantum. Thus, the model probably determined during training that quantumness was unnecessary for this type of task. The conditional-GQE workflow remains applicable to quantum problems, and it is still feasible to obtain quantum circuit generators exhibiting quantumness. However, training generators for such problems entails a more intricate cost landscape, making it challenging to train using simple gate pools or vanilla DPO loss, as done in this study. Future research focusing on more carefully designed workflows would therefore be promising, including the incorporation of quantum-specific metrics, such as entanglement entropy, into the loss function.

2.7 Solving combinatorial optimization using a quantum device

At the end of the performance analyses of GQCO, we examined its behavior on a physical quantum device. The target problem was the 10-variable Max-Cut problem illustrated in Fig. 7a. For comparison, we used a two-layer QAOA circuit whose parameters were optimized with a classical simulator. The resulting circuits (Fig. 7b) were then executed on the IonQ Aria quantum processor. Fig. 7c presents the sampling results for varying numbers of shots alongside the state vector computed by the classical simulator.

A key characteristic of GQCO-generated circuits is that resulting quantum state exhibits a distinct observation probability peak at a single computational basis state. In contrast, because QAOA discretely approximates time evolution from a uniform superposition, its resulting quantum state is more complex and less likely to yield a clear peak, particularly when the circuit depth is limited. Consequently, QAOA required more than 100 shots to identify the correct answer in this study, whereas GQCO was able to find it with just a single shot. This disparity in the number of required shots is expected to grow as the number of qubits increase.

Another notable aspect of GQCO becomes evident in cases where the ground state is degenerate. In principle, our GQCO model cannot account for degeneracy because the training process relies solely on circuit sampling, focusing on identifying a solution without considering the underlying quantum mechanics of the input Ising Hamiltonian. In a Max-Cut problem, the system is inherently degenerate, particularly the target problem in this section is doubly degenerate. As illustrated in Fig. 7c, while GQCO identified only one of the two solution candidates, QAOA exhibited observation probability peaks for both degenerated ground states. Originating in adiabatic quantum computation, QAOA is theoretically expected to yield a non-trivial probability distribution over degenerate ground states. GQCO's inability to capture degeneracy will require future work on model architectures and training approaches. Possible directions include directly incorporating degeneracy-aware constraints into the loss function, embedding symmetries of problems into circuit architectures, or initializing circuits with uniform superpositions.

3 Discussion

We developed a novel quantum-classical hybrid technique for context-aware quantum circuit generation and then applied it to combinatorial optimization problems. Our approach, which we have named conditional-GQE, extends GQE by integrating contextual encoding, and employs the cutting-edge methodologies such as DPO-based training and qubit-based curriculum learning to yield a scalable workflow. This strategy enabled us to successfully build the quantum circuit generator for combinatorial optimization, a high-performance solver that outperforms conventional solvers for problems with up to 10 variables. Although this work is still a prototype, the results suggest the potential for more practical, larger-scale implementations toward foundation models of combinatorial optimization. Moreover, we highlight the capacity of classical neural networks to generate flexible, high-quality quantum circuits, paving the way for advanced quantum-classical hybrid technologies.

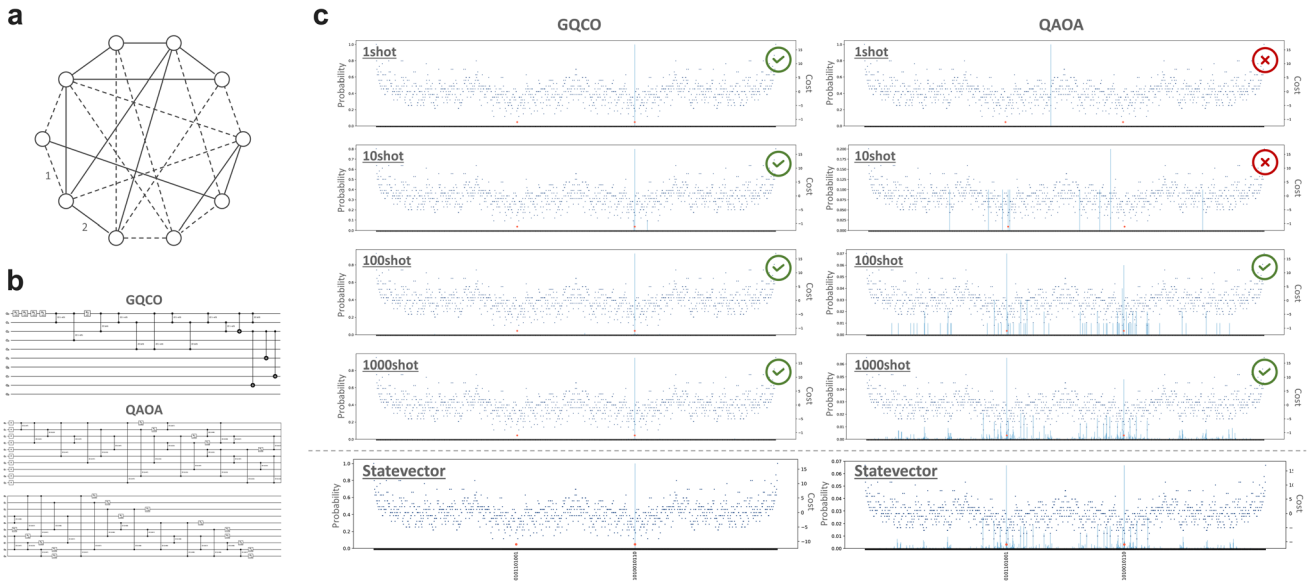

Fig. 7 Results on the real quantum processor. (a) Target Max-Cut problem with 10 variables. The edge weights are represented by line styles: dashed lines indicate a weight of 1, and solid lines indicate a weight of 2. (b) Quantum circuits generated by GQCO and a two-layer QAOA circuit for the target problem. (c) Sampling results on the real quantum device (IonQ Aria) for each of the circuits. Results for 1, 10, 100, and 1000 shots, as well as the state vectors computed by the simulator, are included. The histograms and plots are interpreted in the same manner as in Fig. 5c. Each plot is marked to indicate whether it leads to the correct solution. The enlarged figures are available in the ESI.†

The conceptual workflow described in this study can be extended beyond combinatorial optimization to any problem formulated as observable expectation value minimization. For example, in molecular ground-state searches, representing molecular structures as graphs allows direct adoption of our graph-based encoding. By replacing the encoder, the GQE-based approach also generalizes to quantum machine learning and partial differential equation solvers. We thus view GQE-based quantum circuit generation as a next step following VQAs.

However, several limitations remain as open problems. A major obstacle is the significant classical computational resources required to achieve scalability. While our findings indicate the computational advantage over brute-force solvers and QAOA, fully realizing this advantage demands extensive classical training beforehand. Fig. 8 illustrates the average computational time required for quantum circuit simulations during one epoch of GQCO model training, as well as the proportion of this simulation time relative to the average total computational time per epoch. The comparison spans various number of qubits and contrasts CPU and GPU simulation performance. A significant portion of the GQCO model's training time was consumed by quantum circuit simulations, and this computational cost increases exponentially with the number of qubits. Utilizing high-performance computing resources, such as GPUs, can help mitigate this rapid growth in computational cost.[10] However, classical simulations become impractical for quantum circuits exceeding approximately 50 qubits, rendering conditional-GQE model training infeasible in such scenarios. Direct integration of quantum computations into the training process thus becomes necessary in these regimes, although this inevitably introduces new challenges, including training instability arising from sampling randomness and a substantial increase in the number of shots required.

Furthermore, enhancing the efficiency of generator training and reducing the required number of training epochs are essential objectives. This can be achieved by refining training strategies, such as designing encoder architectures informed by domain knowledge and developing effective pre-training methods. Additionally, careful gate pool design will play a crucial role. Machine learning-based approaches for identifying suitable gates or gate representation learning may offer a promising direction.

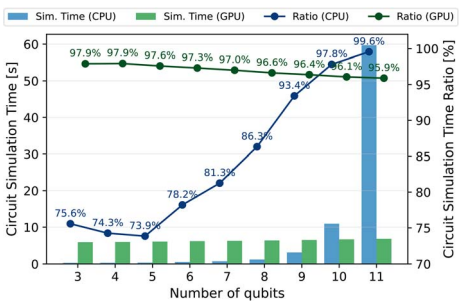

Fig. 8 Average computational time per epoch for quantum circuit simulations, and their proportion relative to the average total computational time per epoch. Blue bars represent the average quantum circuit simulation time using CPUs, while green bars represent the simulation time using GPUs. Line plots indicate the proportion of simulation time relative to the average total training time per epoch, including all steps from generating training data to updating model parameters. In both CPU and GPU simulation scenarios, gradient computations for model parameter updates are conducted on GPUs.

This research provides a novel pathway for quantum computation by leveraging large-scale machine learning models. It underscores the growing role of AI in the advancement of next-generation quantum computing research activities. We believe that our work will serve as a catalyst for accelerating the development of quantum applications across diverse domains and facilitating the democratization of quantum technology.

4 Methods

4.1 GQE, GPT-QE, and conditional-GQE

Let $t = \{t_1, \ldots, t_N\}$ be a generated token sequence of length N, where each token index t_k is an integer satisfying $1 \leq t_k \leq V$, with V being the vocabulary size. Each token index t_k corresponds to a quantum circuit component U_k selected from a predefined operator pool $\{U_\ell\}_{\ell=1}^{V}$. These components collectively form a quantum circuit $U = U_{t_N} \cdots U_{t_1}$. Let $p_\theta(U)$ denote the generative model of quantum circuits, where $p_\theta(U)$ is a probability distribution over unitary operators U, and θ is the set of optimizable parameters. In GQE, the parameters θ are iteratively optimized so that circuits sampled from $p_\theta(U)$ are more likely to minimize the expectation value of an observable:

$$\langle \mathcal{O} \rangle_U := \langle \phi_{\text{ini}} | U^\dagger \mathcal{O} U | \phi_{\text{ini}} \rangle$$

where $\mathcal{O}$ is an observable and $|\phi_{\text{ini}}\rangle$ is a fixed input state. In particular, for an n qubit system, we use $|\phi_{\text{ini}}\rangle = |0\rangle^{\otimes n}$. The quantum computation is involved in the estimation of $\langle \mathcal{O} \rangle_U$. Notably, unlike in VQAs, all optimizable parameters are embedded in the classical generative model $p_\theta(U)$ rather than in the quantum circuit itself (see Fig. 1b).

As discussed in the Result section, the observable can be expressed as a function of certain variables x. Let us denote such an observable as $\mathcal{O}(x)$. The quantum circuit U that minimizes $\langle \mathcal{O}(x) \rangle_U$ also depends on the variable x. In original GQE approach (including GPT-QE), parameters are set and optimized for each specific target problem, much like in VQAs. More precisely, GPT-QE aims to obtain a decoder-only transformer $p_{\theta^*(x)}(U)$ for each x, where $\theta^*(x)$ is the solution for the following minimization problem:

$$\theta^*(x) = \operatorname*{argmin}_\theta \mathop{\mathbb{E}}_{U \sim p_\theta(U)} \langle \mathcal{O}(x) \rangle_U, \tag{1}$$

where $\mathbb{E}_{X \sim p(X)}[f(X)]$ denotes the expectation value of $f(X)$ with respect to the random variable X over the sample space Ω, where X is drawn from the distribution $p(X)$, *i.e.*, $\mathbb{E}_{X \sim p(X)}[f(X)] := \int_\Omega f(X) p(X) \mathrm{d}X$.

By utilizing x as context (*i.e.*, input), conditional-GQE aims to train a generative model $p_\theta(U|x)$ that generates circuits minimizing $\langle \mathcal{O}(x) \rangle_U$. The function $p_\theta(U|x)$ provides the conditional probability of generating the unitary operator U when the input x is given. In transformer-based generative models, the probability $p_\theta(U|x)$ is expressed as follows:

$$p_\theta(U|x) = \prod_{i=1}^{N} p_\theta\left(U_{t_i} | U_{t_0}, \ldots, U_{t_{i-1}}, x\right) \propto \prod_{i=1}^{N} \exp\left\{ \frac{z_i\left(U_{t_0}, \ldots, U_{t_{i-1}}, x; \theta\right)}{T} \right\}$$

where t_0 is the start-token index, chosen such that $U_{t_0} = I^{\otimes n}$ in this work. z_i denotes the logit for i-th token, that is, the corresponding output from the model before applying the sigmoid function. T is the sampling temperature; in this work, we set $T = 1.0$ for training and $T = 2.0$ for evaluation, thereby enhancing randomness in the evaluation phase. Then, we can realize a generative model $p_{\theta^*}(U|x)$ through the following optimization:

$$\theta^* = \operatorname*{argmin}_\theta \mathop{\mathbb{E}}_{x \sim p(x)} \mathop{\mathbb{E}}_{U \sim p_\theta(U|x)} \langle \mathcal{O}(x) \rangle_U \tag{2}$$

where $p(x)$ denotes the probability distribution of inputs x in the target domain.

Solving the optimization problems in eqn (1) or eqn (2) is challenging and thus requires surrogate objective functions. GPT-QE employs a logit-matching approach, whereas this study utilizes DPO[39] loss. Further details of DPO are provided in the subsequent section.

4.2 Outline of the GQCO model architecture

The GQCO model proposed in this study employs an encoder–decoder transformer architecture[32] with GELU activation.[60] Its architectural diagram is provided in the ESI.† Both encoder and decoder consist of 12 repeated modules, each incorporating graph convolution[47] and a multi-head attention transformer. The encoder components are detailed in subsequent sections. The intermediate representations have dimension 256, and each transformer layer has 8 attention heads. In total, the model has approximately 256 million parameters, with 127 million in the encoder and 129 million in the decoder. This parameter count is comparable to T5,[61] an early encoder–decoder LLM with 246 million parameters, and the decoder alone is similar in scale to GPT-1,[62] which is a decoder-only transformer model and has 117 million parameters. Our gate pool supports the generation of quantum circuits with up to 20 qubits and offers 1901 gate candidates, including an identity gate, although we have only proceeded to the 10-qubit scale. During generation, gates unnecessary for the given problem size are masked. The length of the token index sequence, which corresponds to the number of generated gates, is set to a maximum of $2n$, where n is the number of qubits used. Note that, when four or more gates have been generated, and an end-token index $t_{\text{end}} = t_0$ is produced, the generation process is terminated before reaching the maximum length.

4.3 Embedding combinatorial optimization problems as graphs

Any combinatorial optimization problem can be bijectively mapped to an Ising Hamiltonian:[46]

$$\mathcal{H} = \sum_{i<j} J_{ij} \sigma_i^z \sigma \sigma_j^z + \sum_i h_i \sigma_i^z$$

whose ground states correspond to the solutions of the problem. The coefficients of the Ising Hamiltonian—the external magnetic field h_i and the interaction coefficient J_{ij}—are used as inputs (or contexts) to the encoder of the model. We map these coefficients into a graph representation, considering

h_i as the weight of node i and J_{ij} as the weight of the edge between nodes i and j.

The feature vector is then constructed using the following three elements: (1) the weights themselves, (2) the sign of the magnitude relationships between the weights of adjacent nodes or edges, and (3) the sign of the product of the weights of adjacent nodes or edges (see Fig. 2). More formally, the node feature v_i and edge feature e_{ij} are computed as follows:

$$v_i = \begin{bmatrix} v_i^{(1)} \\ v_i^{(2)} \\ v_i^{(3)} \\ v_i^{(4)} \end{bmatrix}, \quad e_{ij} = \begin{bmatrix} \mathrm{sgn}(J_{ij}) \\ \mathrm{sgn}(J_{ij} - h_i) \\ \mathrm{sgn}(J_{ij} - h_j) \\ \mathrm{sgn}(h_i h_j J_{ij}) \end{bmatrix},$$

$$v_i^{(1)} = h_i, \quad v_i^{(2)} = \begin{bmatrix} \mathrm{sgn}(h_i - h_{j_1}) \\ \mathrm{sgn}(h_i - h_{j_2}) \\ \vdots \\ \mathrm{sgn}(h_i - h_{j_k}) \end{bmatrix},$$

$$v_i^{(3)} = \begin{bmatrix} \mathrm{sgn}(h_i - J_{ij_1}) \\ \mathrm{sgn}(h_i - J_{ij_2}) \\ \vdots \\ \mathrm{sgn}(h_i - J_{ij_k}) \end{bmatrix}, \quad v_i^{(4)} = \begin{bmatrix} \mathrm{sgn}(h_i h_{j_1} J_{ij_1}) \\ \mathrm{sgn}(h_i h_{j_2} J_{ij_2}) \\ \vdots \\ \mathrm{sgn}(h_i h_{j_k} J_{ij_k}) \end{bmatrix}$$

where $\mathrm{sgn}(\cdot)$ is the sign function and $\{j_\ell\}_{\ell=1}^{k} (=: \mathcal{N}(i))$ denote the index set of nodes connected to node i. These handcrafted features serve to incorporate domain knowledge of the Ising model into our model; specifically, the facts that spin–spin interactions with large coefficients or strong external magnetic fields have a significant impact on the spin configuration of the system, and that frustration—the absence of a spin configuration that simultaneously minimizes all interaction energies—generates a complex energy landscape.

4.4 Encoder with graph transformer convolution

The embedded graphs are converted into encoded representations by alternately applying graph transformer convolutional layers[47] and feed-forward layers. Specifically, local message passing and feature transformation are performed according to the following equations:

$$v_i' = \mathrm{LayerNorm}\left(W_1 v_i + \sum_{j \in \mathcal{N}(i)} \alpha_{ij} (W_2 v_j + W_3 e_{ij}) \right),$$

$$\alpha_{ij} = \mathrm{softmax}\left(\frac{(W_4 v_i)^\top (W_5 v_j + W_6 e_{ij})}{\sqrt{d}} \right),$$

$$v_i'' = \mathrm{LayerNorm}\left(v_i' + W_8 \mathrm{GELU}\left(W_7 v_i' \right) \right),$$

where the matrices $W_k (k = 1, \ldots, 8)$ are trainable weight, and d is the dimension of the intermediate representations (*i.e.* $d = 256$). LayerNorm$(\cdot)$ is layer normalization,[63] softmax$(\cdot)$ denotes the softmax function, and GELU$(\cdot)$ refers the GELU activation function[60] applied element-wise. This base module is repeated 12 times, and the resulting node features, indexed by node order, serve as input to the decoder. Although using 12 iterations risks issues of graph neural networks like over-smoothing,[64] over-squashing,[65] and graph bottlenecks,[66] we set the iteration count to 12 to align the model scale with that of well-known language models such as T5 and GPT-1.

4.5 Direct preference optimization

Traditionally, supervised learning with labeled dataset has been widely used in machine learning applications to quantum information processing. Examples include quantum circuit compilation,[37,38,67] ansatz generation for VQAs,[68,69] and diffusion-based quantum circuit generation.[35] However, this approach faces some significant challenges. The most notable issue is the scalability. Preparing training data using classical computation quickly becomes infeasible for large-scale circuits exceeding 50 qubits. Furthermore, for complex tasks, it is difficult to prepare ground-truth circuits, or the circuit structure may not be uniquely defined. Consequently, preparing high-quality training datasets remains problematic.

For these reasons, we employ a preference-based approach using direct preference optimization (DPO).[39] DPO is a training strategy derived in the field of reinforcement learning from human feedback (RLHF),[70,71] used to fine-tune LLMs for generating preferred outputs. In this approach, multiple outputs are sampled and parameters are updated to increase the likelihood of preferred outputs while decreasing that of less preferred ones. Typically, in LLMs, human evaluators determine the preference of outputs. In our study, we assess the preferrability of circuits using the computed expectation values of the Hamiltonian.

Fig. 9a shows the schematics of our DPO-based training process. The expected DPO loss function used in this work is defined by:

$$\mathscr{L}_{\mathrm{DPO}}(\theta) := \mathop{\mathbb{E}}_{x} \mathop{\mathbb{E}}_{w \succ \ell} L(w, \ell, x; \theta),$$

$$L(w, \ell, x; \theta) : = \log\left\{ 1 + \exp\left\{ -\beta \left(\log \frac{p_\theta(U^{(w)}|x)}{\pi_{\mathrm{ref}}(U^{(w)}|x)} - \log \frac{p_\theta(U^{(\ell)}|x)}{\pi_{\mathrm{ref}}(U^{(\ell)}|x)} \right) \right\} \right\}$$

where $\{U^{(k)}\}_{k=1}^{\mathrm{M}}$ is the set of sampled circuits, and $w \succ \ell$ indicates that $\langle \mathscr{O}(x) \rangle_{U^{(w)}} < \langle \mathscr{O}(x) \rangle_{U^{(\ell)}}$, *i.e.*, the circuit $U^{(w)}$ is preferred over the circuit $U^{(\ell)}$. $\pi_{\mathrm{ref}}(\cdot|x)$ is a reference probability and serves as the baseline for the optimization during the DPO, and β is a hyperparameter controlling the influence of π_{ref}. In this work, we use $\pi_{\mathrm{ref}}(U|x) \propto \exp\{ -\langle \mathscr{O}(x) \rangle_U \}$. Since the negative log-sigmoid function $\log(1 + \exp(-x))$ is monotonically

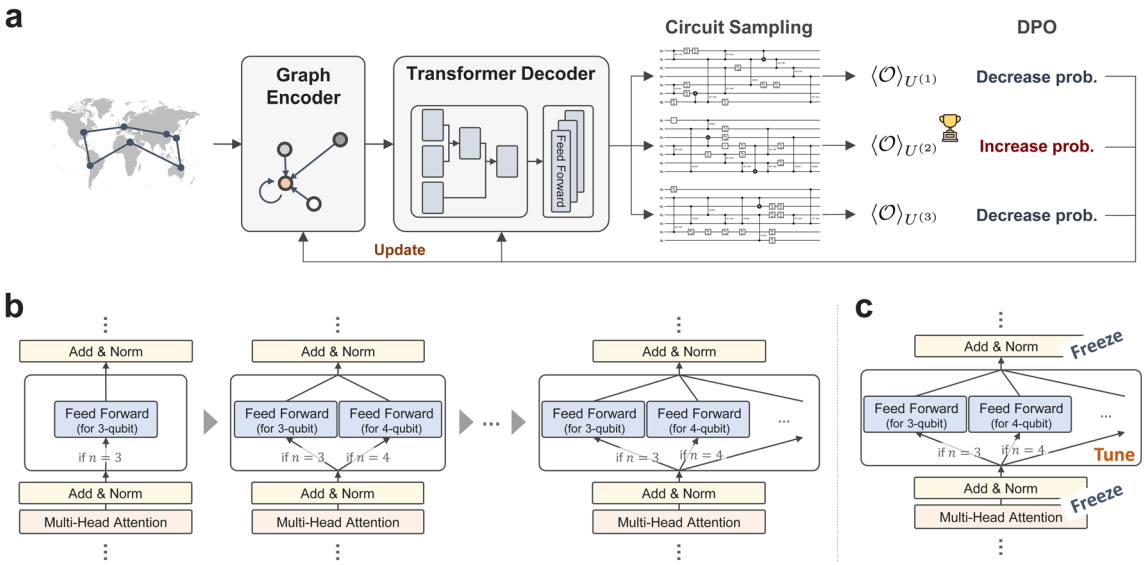

Fig. 9 Training strategy for GQCO. (a) Training iterations of direct preference optimization (DPO). For a randomly generated input problem, multiple quantum circuits are sampled, and the expected value of the Hamiltonian is computed for each one. The model parameters of the encoder and decoder are updated to increase the probability of generating the circuit with the lowest energy value while decreasing the probabilities of generating the other circuits. (b) Curriculum learning based on the number of qubits. We start training by generating circuits with three qubits and gradually increase the complexity of the task by increasing the number of qubits. (c) Expert tuning. The weights of the shared layers, such as the attention layers, are fixed, and the expert layers are fine-tuned.

decreasing, the function L is minimized when $p_\theta(U^{(w)}|x)$ is maximized and $p_\theta(U^{(\ell)}|x)$ is minimized. In other words, this loss function is designed to increase the generation probability of preferred circuits while decrease the probability of non-preferred circuits.

Ideally, the function L should be computed for all possible pairs of the M sampled circuits, totaling $M(M-1)/2$ combinations. However, to reduce computational overhead, we employ the best-vs-others empirical loss, defined as follows:

$$\mathscr{L}_{\mathrm{DPO}}^{\mathrm{BvO}}(\theta) := \frac{1}{|\mathscr{D}_x|}\sum_{x\in\mathscr{D}_x}\frac{1}{M-1}\sum_{\ell}L(w_{\mathrm{best}},\ell,x;\theta)$$

where w_{best} represents the index of the circuit with the smallest expectation value for each input x, and $\mathscr{D}_x$ is an input dataset with size $|\mathscr{D}_x|$.

However, if all M sampled circuits are identical, the gradient of the loss will be zero regardless of the magnitude of the expected values, preventing the model from being trained effectively. To mitigate this issue, we employ contrastive preference optimization (CPO),[72] an improved version of DPO. In addition to the DPO loss, CPO introduces a negative log-likelihood term on the probability of generating the most preferred output. In summary, the loss function used in this work is given by:

$$\mathscr{L}_{\mathrm{CPO}}^{\mathrm{BvO}}(\theta) := \frac{1}{|\mathscr{D}_x|}\sum_{x\in\mathscr{D}_x}\left[\frac{1}{M-1}\sum_{\ell}L(w_{\mathrm{best}},\ell,x;\theta) - p_\theta\left(U^{(w_{\mathrm{best}})}|x\right)\right] \tag{3}$$

The hyperparameter β is set to 0.1, and the number of samplings M is adjusted according to the number of qubits to maximize the utilization of computational memory. Model training proceeds by generating an input x uniformly at random as a coefficient of the Ising Hamiltonian (*i.e.*, $|\mathscr{D}_x| = 1$), computing the gradient based on the loss function eqn (3), and iteratively updating the parameters. See the ESI† for detailed training settings, including the values of M.

4.6 Qubit-based mixture-of-experts

Depending on the number of variables in a combinatorial optimization problem, quantum computation requires a corresponding number of qubits. To effectively handle the diversity of tasks resulting from variations in problem size, we employ the mixture-of-experts (MoE) architecture,[49–51] which is commonly used in LLMs. As illustrated in Fig. 2 and 9b, c, each feed-forward module in the model is partitioned into specialized submodules, referred to as "experts". The gating mechanism dynamically selects layers based on the number of qubits, forming what we term a qubit-based MoE. This design balances the need for diverse model representations while limiting the growth of model parameters.

4.7 Curriculum learning

The qubit-based MoE enhances model scalability. By incorporating additional experts and restarting training, circuit generators can be trained for varying numbers of qubits without the need to retrain the entire model from scratch. Leveraging this capability, we adopt curriculum learning,[52] a method that incrementally increases task complexity by starting with simpler problems and gradually progressing to more challenging ones.

Training begins with randomly generated combinatorial optimization problems involving 3 qubits. Performance is

monitored regularly, and training continues until the model achieves an accuracy exceeding 90% on randomly generated test problems. Once this threshold is met, size-4 optimization problems are introduced as training candidates, along with the integration of a new expert module within the MoE layers. Then, performance is continuously monitored, and the maximum problem size in the training dataset is gradually increased.

Even when the maximum problem size is $n_{\max}$, problems involving fewer qubits ($<n_{\max}$) are still generated as part of the training data. The probability of generating a problem of size n when the current maximum size is $n_{\max}$ is defined as:

$$p(n|n_{\max}) = \begin{cases} 0 & \text{if } n \leq n_{\max} < 3 \\ 1 & \text{if } n = n_{\max} = 3, \\ \dfrac{0.5}{n_{\max} - 3} & \text{if } 3 \leq n < n_{\max}, \\ 0.5 & \text{if } 3 < n = n_{\max}. \end{cases} \quad (4)$$

In brief, when the number of qubits is larger than three, the probability of generating the largest problem size is 0.5, and the remaining 0.5 probability is equally divided among all smaller sizes. Notably, the probability of generating previously trained problem sizes is not set to zero. This strategy mitigates catastrophic forgetting[73]—a phenomenon in which performance on previously learned tasks declines rapidly in continuous learning and online learning. Without such adjustments, continuously updating training data could lead to the model forgetting previously acquired knowledge. A small amount of instances for smaller problem sizes helps to maintain consistent performance across all problem sizes.

4.8 Simulated annealing

In this study, simulated annealing (SA) was implemented using D-Wave's Ocean library. The number of reads per sampling was fixed at 100, while the number of sweeps varied across the set $\{10^2, 10^3, 10^4, 10^5, 10^6, 10^7\}$ to evaluate the performance.

4.9 Quantum approximate optimization algorithm (QAOA)

The standard QAOA[34] was employed as the baseline for the quantum combinatorial optimization solver. The number of layers was varied from 1 to 4. Specifically, the quantum circuit was initialized by applying Hadamard gates to all qubits, followed by repeated applications (from 1 to 4 iterations) of the problem Hamiltonian and mixing Hamiltonian. Parameter optimization was conducted using the Nelder–Mead algorithm,[74] with initial parameters uniformly and randomly initialized within the range $-\pi/8$ to $\pi/8$. Optimization iterations were capped at a maximum of 1000 steps. All quantum circuit construction, simulation, and parameter optimization were performed using CUDA-Q[75] on GPU hardware.

4.10 Hardware configuration

Multiple compute nodes, each consisting of four NVIDIA V100 GPUs, were used, and the model was trained using a distributed data parallel (DDP) strategy.[76] The quantum circuit simulations were performed on a CPU (Intel Xeon Gold 6148 processor) environment using Qiskit.[57] The number of GPU nodes used in training and the epochs differ for each stage of the curriculum learning. The details are summarized in the ESI.†

For performance evaluation, the model inferences and quantum circuit simulations were both performed on an NVIDIA RTX A6000 GPU. All other classical computations were performed on an Intel Core i9-14900K CPU. For the real quantum device, we used IonQ Aria *via* Amazon Braket.

Data availability

All training datasets were randomly generated during training and were therefore not stored. Instead, the random seeds are fixed to ensure reproducibility. These seeds, along with the test dataset and code, are available at **https://github.com/shunyaist/generative-quantum-combinatorial-optimization** and archived at **https://doi.org/10.5281/zenodo.15859123**. The trained and finetuned models are available at **https://doi.org/10.5281/zenodo.15858977**.

Author contributions

AAG, KN, and TK conceived the conceptual ideas of this study. SM, YS, and TK deviced and outlined the project. SM implemented the machine-learning algorithms and conducted the experiments with the support of KN, YS, and TK. SM and KN wrote the manuscript, and all authors have edited, read and approved the final manuscript.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

This work was performed for Council for Science, Technology and Innovation (CSTI), Cross-ministerial Strategic Innovation Promotion Program (SIP), "Promoting the application of advanced quantum technology platforms to social issues" (Funding agency : QST). This work was supported by JSPS KAKENHI Grant-in-Aid for Research Activity Start-up (23K19980). Classical computational resources were provided by AI Bridging Cloud Infrastructure (ABCI) of National Institute of Advanced Industrial Science and Technology (AIST). Quantum computational resources were provided by Amazon Braket. SM, YS, and TK would like to express the gratitude to Yusuke Hama and Shiro Kawabata for their insightful discussions and supports. AAG thanks Anders G. Frøseth for his generous support and acknowledges the generous support of Natural Resources Canada and the Canada 150 Research Chairs program.

References

1 C. Ryan-Anderson, N. Brown, M. Allman, B. Arkin, G. Asa-Attuah, C. Baldwin, J. Berg, J. Bohnet, S. Braxton,

N. Burdick *et al.*, *arXiv*, 2022, preprint, arXiv:2208.01863, DOI: 10.48550/arXiv.2208.01863.
2 V. Sivak, A. Eickbusch, B. Royer, S. Singh, I. Tsioutsios, S. Ganjam, A. Miano, B. Brock, A. Ding, L. Frunzio, *et al.*, *Nature*, 2023, **616**, 50–55.
3 D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. Bonilla Ataides, N. Maskara, I. Cong, X. Gao, P. Sales Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletić and M. D. Lukin, *Nature*, 2024, **626**, 58–65.
4 H. Putterman, K. Noh, C. T. Hann, G. S. MacCabe, S. Aghaeimeibodi, R. N. Patel, M. Lee, W. M. Jones, H. Moradinejad and R. Rodriguez, *Nature*, 2025, **638**, 927–934.
5 B. W. Reichardt, D. Aasen, R. Chao, A. Chernoguzov, W. van Dam, J. P. Gaebler, D. Gresh, D. Lucchetti, M. Mills, S. A. Moses, B. Neyenhuis, A. Paetznick, A. Paz, P. E. Siegfried, M. P. da Silva, K. M. Svore, Z. Wang and M. Zanner, *arXiv*, 2024, preprint, arXiv:2409.04628, DOI: 10.48550/arXiv.2409.04628.
6 R. Acharya, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev, J. Atalaya, *et al.*, *Nature*, 2024, **638**, 920–926.
7 S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall and T. J. Yoder, *Nature*, 2024, **627**, 778–782.
8 Q. Xu, J. P. Bonilla Ataides, C. A. Pattison, N. Raveendran, D. Bluvstein, J. Wurtz, B. Vasić, M. D. Lukin, L. Jiang and H. Zhou, *Nat. Phys.*, 2024, 1–7.
9 K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, *et al.*, *Rev. Mod. Phys.*, 2022, **94**, 015004.
10 Y. Alexeev, M. H. Farag, T. L. Patti, M. E. Wolf, N. Ares, A. Aspuru-Guzik, S. C. Benjamin, Z. Cai, Z. Chandani, F. Fedele, N. Harrigan, J.-S. Kim, E. Kyoseva, J. G. G. Lietz, T. Lubowe, A. McCaskey, R. G. Melko, K. Nakaji, A. Peruzzo, S. Stanwyck, N. M. Tubman, H. Wang and T. Costa, *arXiv*, 2024, preprint, arXiv:2411.09131, DOI: 10.48550/arXiv.2411.09131.
11 A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik and J. L. O'brien, *Nat. Commun.*, 2014, **5**, 4213.
12 J. R. McClean, J. Romero, R. Babbush and A. Aspuru-Guzik, *New J. Phys.*, 2016, **18**, 023023.
13 M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, *Nat. Rev. Phys.*, 2021, **3**, 625–644.
14 K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek and A. Aspuru-Guzik, *Rev. Mod. Phys.*, 2022, **94**, 015004.
15 S. Lloyd, M. Mohseni and P. Rebentrost, *arXiv*, 2013, preprint, arXiv:1307.0411, DOI: 10.48550/arXiv.1307.0411.
16 I. S. Maria Schuld and F. Petruccione, *Contemp. Phys.*, 2015, **56**, 172–185.
17 J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe and S. Lloyd, *Nature*, 2017, **549**, 195–202.
18 K. Mitarai, M. Negoro, M. Kitagawa and K. Fujii, *Phys. Rev. A*, 2018, **98**, 032309.
19 A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster and J. I. Latorre, *Quantum*, 2020, **4**, 226.
20 F. J. Gil Vidal and D. O. Theis, *Front. Phys.*, 2020, **8**, 297.
21 M. Schuld, R. Sweke and J. J. Meyer, *Phys. Rev. A*, 2021, **103**, 032430.
22 J. Kübler, S. Buchholz and B. Schölkopf, *Adv. Neural Inf. Process. Syst.*, 2021, **34**, 12661–12673.
23 J. Baxter, *J. Artif. Intell. Res.*, 2000, **12**, 149–198.
24 K. Nakaji, L. B. Kristensen, J. A. Campos-Gonzalez-Angulo, M. G. Vakili, H. Huang, M. Bagherimehrab, C. Gorgulla, F. Wong, A. McCaskey, J.-S. Kim, T. Nguyen, P. Rao and A. Aspuru-Guzik, *arXiv*, 2024, preprint, arXiv:2401.09253, DOI: 10.48550/arXiv.2401.09253.
25 A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, *OpenAI blog*, 2019, **1**, 9.
26 A. Krizhevsky, I. Sutskever and G. E. Hinton, *Adv. Neural Inf. Process. Syst.*, 2012, **25**(2), DOI: 10.1145/3065386.
27 K. Simonyan and A. Zisserman, *arXiv*, 2014, preprint, arXiv:1409.1556, DOI: 10.48550/arXiv.1409.1556.
28 K. He, X. Zhang, S. Ren and J. Sun, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, 2016.
29 D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik and R. P. Adams, *Advances in Neural Information Processing Systems*, 2015.
30 S. Kearnes, K. McCloskey, M. Berndl, V. Pande and P. Riley, *J. Comput. Aided Mol. Des*, 2016, **30**, 595–608.
31 P. Reiser, M. Neubert, A. Eberhard, L. Torresi, C. Zhou, C. Shao, H. Metni, C. van Hoesel, H. Schopmans, T. Sommer, *et al.*, *Commun. Mater.*, 2022, **3**, 93.
32 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser and I. Polosukhin, *Advances in Neural Information Processing Systems*, 2017.
33 R. Sato, *arXiv*, 2020, preprint, arXiv:2003.04078, DOI: 10.48550/arXiv.2003.04078.
34 E. Farhi, J. Goldstone and S. Gutmann, *arXiv*, 2014, preprint, arXiv:1411.4028, DOI: 10.48550/arXiv.1411.4028.
35 F. Fürrutter, G. Muñoz-Gil and H. J. Briegel, *Nat. Mach. Intell.*, 2024, 1–10.
36 S. Daimon and Y. Matsushita, *Phys. Rev. Appl.*, 2024, **22**, L041001.
37 Y.-H. Zhang, P.-L. Zheng, Y. Zhang and D.-L. Deng, *Phys. Rev. Lett.*, 2020, **125**, 170501.
38 F. Preti, M. Schilling, S. Jerbi, L. M. Trenkwalder, H. P. Nautrup, F. Motzoi and H. J. Briegel, *Quantum*, 2024, **8**, 1343.
39 R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon and C. Finn, *Adv. Neural Inf. Process. Syst.*, 2023, 53728–53741.
40 S. Kirkpatrick, C. D. Gelatt Jr and M. P. Vecchi, *science*, 1983, **220**, 671–680.
41 M. Gasse, S. Bowly, Q. Cappart, J. Charfreitag, L. Charlin, D. Chételat, A. Chmiela, J. Dumouchelle, A. Gleixner, A. M. Kazachkov *et al.*, *NeurIPS 2021 competitions and demonstrations track*, 2022, pp. 220–231.

42 F. Berto, C. Hua, J. Park, L. Luttmann, Y. Ma, F. Bu, J. Wang, H. Ye, M. Kim, S. Choi, N. G. Zepeda, A. Hottung, J. Zhou, J. Bi, Y. Hu, F. Liu, H. Kim, J. Son, H. Kim, D. Angioni, W. Kool, Z. Cao, J. Zhang, K. Shin, C. Wu, S. Ahn, G. Song, C. Kwon, L. Xie and J. Park, *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.
43 T. Kadowaki and H. Nishimori, *Phys. Rev. E*, 1998, **58**, 5355.
44 J. Alcazar, M. Ghazi Vakili, C. B. Kalayci and A. Perdomo-Ortiz, *Nat. Commun.*, 2024, **15**, 2761.
45 A. Cervera-Lierta, J. S. Kottmann and A. Aspuru-Guzik, *PRX Quantum*, 2021, **2**, 020329.
46 A. Lucas, *Front. Phys.*, 2014, **2**, 5.
47 Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang and Y. Sun, *Proceedings of IJCAI-21*, 2021, pp. 1548–1554.
48 P. K. Barkoutsos, J. F. Gonthier, I. Sokolov, N. Moll, G. Salis, A. Fuhrer, M. Ganzhorn, D. J. Egger, M. Troyer, A. Mezzacapo, S. Filipp and I. Tavernelli, *Phys. Rev. A*, 2018, **98**, 022322.
49 R. A. Jacobs, M. I. Jordan, S. J. Nowlan and G. E. Hinton, *Neural Comput.*, 1991, **3**, 79–87.
50 N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton and J. Dean, *International Conference on Learning Representations*, ICLR, 2017.
51 W. Fedus, B. Zoph and N. Shazeer, *J. Mach. Learn. Res.*, 2022, **23**, 1–39.
52 P. Soviany, R. T. Ionescu, P. Rota and N. Sebe, *Int. J. Comput. Vis.*, 2022, **130**, 1526–1565.
53 A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. N. Gomez, S. Gouws, L. Jones, Ł. Kaiser, N. Kalchbrenner, N. Parmar *et al.*, *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas*, 2018, pp. 193–199.
54 L. Chen, J. Q. Davis, B. Hanin, P. Bailis, I. Stoica, M. Zaharia and J. Zou, *Adv. Neural Inf. Process. Syst.*, 2025, 45767–45790.
55 B. Brown, J. Juravsky, R. Ehrlich, R. Clark, Q. V. Le, C. Ré and A. Mirhoseini, *arXiv*, 2024, preprint, arXiv:2407.21787, DOI: **10.48550/arXiv.2407.21787**.
56 C. Snell, J. Lee, K. Xu and A. Kumar, *The Thirteenth International Conference on Learning Representations*, 2025.
57 A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson and J. M. Gambetta, *arXiv*, 2024, preprint, arXiv:2405.08810, DOI: **10.48550/arXiv.2405.08810**.
58 F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, *Nature*, 2019, **574**, 505–510.
59 Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. Van Den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, *et al.*, *Nature*, 2023, **618**, 500–505.
60 D. Hendrycks and K. Gimpel, *arXiv*, 2016, preprint, arXiv:1606.08415, DOI: **10.48550/arXiv.1606.08415**.
61 C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, *J. Mach. Learn. Res.*, 2020, **21**, 1–67.
62 A. Radford, *Improving language understanding by generative pre-training*, 2018, **https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf**.
63 J. Lei Ba, J. R. Kiros and G. E. Hinton, *arXiv*, 2016, preprint, arXiv:1607.06450.
64 Q. Li, Z. Han and X.-M. Wu, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, 2018.
65 J. Topping, F. D. Giovanni, B. P. Chamberlain, X. Dong and M. M. Bronstein, *International Conference on Learning Representations*, 2022.
66 U. Alon and E. Yahav, *International Conference on Learning Representations*, 2021.
67 L. Moro, M. G. Paris, M. Restelli and E. Prati, *Commun. Phys.*, 2021, **4**, 178.
68 S.-X. Zhang, C.-Y. Hsieh, S. Zhang and H. Yao, *Mach. learn. sci. technol.*, 2021, **2**, 045027.
69 Z. He, X. Zhang, C. Chen, Z. Huang, Y. Zhou and H. Situ, *Quant. Inf. Process.*, 2023, **22**, 128.
70 D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano and G. Irving, *arXiv*, 2019, preprint, arXiv:1909.08593.
71 L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, *et al.*, *Adv. Neural Inf. Process. Syst.*, 2022, **35**, 27730–27744.
72 H. Xu, A. Sharaf, Y. Chen, W. Tan, L. Shen, B. V. Durme, K. Murray and Y. J. Kim, *ICML*, 2024, **https://openreview.net/forum?id=51iwkioZpn**.
73 J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, *Proc. Natl. Acad. Sci.*, 2017, **114**, 3521–3526.
74 J. A. Nelder and R. Mead, *Comput. J.*, 1965, **7**, 308–313.
75 The CUDA-Q development team, CUDA-Q, **https://github.com/NVIDIA/cuda-quantum**, Version 1.2.0, Apache-2.0 License.
76 S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania and S. Chintala, *Proc. VLDB Endow.*, 2020, **13**, 3005–3018.