TreeVQA: A Tree-Structured Execution Framework for Shot Reduction in Variational Quantum Algorithms

ASPLOS 2026 camera-ready WIP – Do NOT Distribute!!

Yuewen Hou
isaachyw@umich.edu
University of Michigan
Ann Arbor, MI, USA

Dhanvi Bharadwaj
dhanvib@umich.edu
University of Michigan
Ann Arbor, MI, USA

Gokul Subramanian Ravi
gsravi@umich.edu
University of Michigan
Ann Arbor, MI, USA

Abstract

Variational Quantum Algorithms (VQAs) are promising for near- and intermediate-term quantum computing, but their execution cost is substantial. Each task requires many iterations and numerous circuits per iteration, and real-world applications often involve multiple tasks, scaling with the precision needed to explore the application's energy landscape. This demands an enormous number of execution shots, making practical use prohibitively expensive.

We observe that VQA costs can be significantly reduced by exploiting execution similarities across an application's tasks. Based on this insight, we propose TreeVQA[1], a tree-based execution framework that begins by executing tasks jointly and progressively branches only as their quantum executions diverge.

Implemented as a VQA wrapper, TreeVQA integrates with typical VQA applications. Evaluations on scientific and combinatorial benchmarks show shot count reductions of 25.9× on average and over 100× for large-scale problems at the same target accuracy. The benefits grow further with increasing problem size and precision requirements.

1 Introduction

Quantum computers are gradually transitioning from near-term noisy intermediate-scale quantum (NISQ) systems [49], which feature at most a few hundred qubits with limited coherence times and high gate error rates, to intermediate-term Early Fault Tolerant (EFT) systems [22, 30], expected to host thousands of qubits and employ limited forms of quantum error correction to reduce error rates. These systems will be unable to run long-term quantum applications [25, 58], which demand millions of qubits, full fault tolerance, and the ability to execute billions of quantum operations [47]. Despite these limitations, the community remains optimistic that near- and intermediate-term devices can deliver practically useful quantum advantage in key domains such as optimization [13], physics [36], and chemistry [48].

One promising class of algorithms with potential feasibility and utility before full fault tolerance is Variational Quantum Algorithms (VQAs). Their inherent robustness [64] to noise allows them to deliver useful results even without full-fledged quantum error correction. VQAs have broad applicability, including estimating the energies of molecules [48] and approximating solutions to optimization problems such as MaxCut [13]. These hybrid algorithms combine a quantum circuit with parameterized angles (*ansatz*) with a classical optimizer. Through an iterative feedback loop, the VQA explores the solution space and converges to the problem's "ground state", representing the optimal solution.

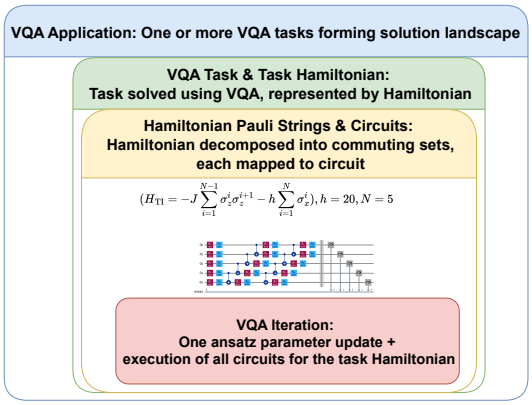

Figure 1. Key terminologies and their relations.

Before delving into VQA details and our proposed work, we introduce key terminology for context: *(i) VQA Application:* An application typically composed of one or more VQA tasks. When multiple tasks are involved, their solutions are combined to construct a solution landscape relevant to the application; *(ii) VQA Task and Task Hamiltonian:* A VQA task is solved using a VQA to find its ground-state solution and is mathematically represented by a Hamiltonian; *(iii) Hamiltonian Pauli String and Circuit:* For a specific VQA task, its Hamiltonian consists of multiple Pauli strings, which can be grouped into sets that fully commute within a set but not across sets. Each set corresponds to a quantum circuit (using the same ansatz but different measurement bases); *(iv) VQA Iteration:* For a specific VQA task, this is a single instance of ansatz parameter updates, as determined by the classical optimizer, followed by the execution of circuits associated with the task Hamiltonian. Fig.1 illustrates this terminology.

While VQAs hold promise, their execution faces signifi-

[1]TreeVQA is open source at https://github.com/isaachyw/TreeVQA.

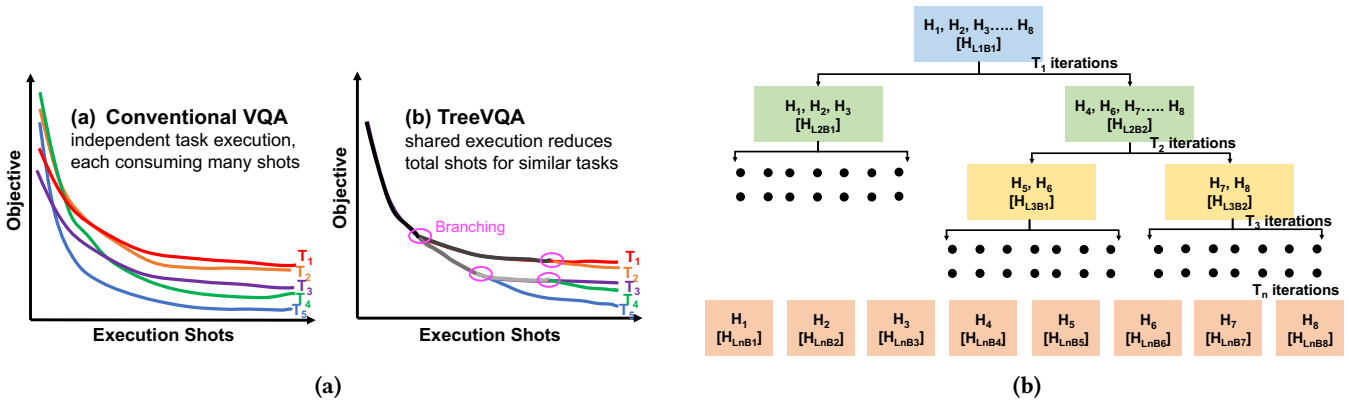

Figure 2. (a) Comparing the iterative execution of conventional VQA and TreeVQA. (b) TreeVQA strategy overview, branching from the root node to multiple leaf nodes for an eight-VQA task application.

cant hurdles, the most critical being the enormous number of 'shots' (executions) required for any real-world application. This challenge arises from three factors: ⓐ VQA tasks are inherently iterative, often requiring *thousands* of iterations to explore the solution landscape of complex quantum problems, a process further complicated by device noise; ⓑ For practical applications such as chemistry, task Hamiltonians contain *thousands* of Pauli terms, resulting in a large number of circuits per iteration; and ⓒ While the above challenges were associated with individual tasks, real-world applications typically comprise *thousands* of VQA tasks, whose collective ground states define the application's solution landscape. Together, these factors drive the total number of circuit executions into the *billions* for practical VQA applications, making time and resource costs a show-stopper.

Prior research, both specific to VQA and otherwise, has contributed to both reducing and exacerbating total shot costs, depending on their primary goal. For instance, on the one hand, classical initialization methods [52, 56, 57] can help reduce the number of VQA iterations, although their success is often limited and problem dependent. On the other hand, error mitigation strategies such as ZNE [18, 34], PEC [3], measurement error mitigation [11], and more, all enormously increase the shot counts. The key takeaway is that, despite prior strategies or because of them, the cost of execution shots is still extremely high and deserves attention.

In this work, we make the novel observation that many VQA tasks within an application are highly similar. In particular, the solutions to these tasks (i.e., the quantum states corresponding to their ground states) often exhibit significant overlap. The key idea is that, while these solutions are distinct enough to require individual evaluation for constructing the high-precision solution landscape demanded by the application, their substantial overlaps create an opportunity to share quantum executions across tasks as they iterate toward their respective solutions. By reusing large portions of the VQA process, the total cost of quantum execution can be reduced dramatically.

Based on this insight, we propose *TreeVQA*, a tree-based execution framework that begins by executing all application VQA tasks as a single (or minimal set of) quantum executions. These executions then gradually branch into multiple distinct paths as the quantum states produced by the VQA process converge toward each task's solution. A comparison of the iterative execution in TreeVQA and conventional VQA is illustrated in Fig.2a. In TreeVQA, multiple VQA tasks from an application (five in the figure: T_1 - T_5) begin by executing as a single entity and then eventually diverge. Dissimilar tasks branch out earlier, while similar tasks stick together until closer to convergence. In comparison, the execution cost is much greater for conventional VQA, which executes all of these tasks entirely individually.

The tree execution strategy employed by TreeVQA is more specifically illustrated in Fig.2b, which shows an example application with eight VQA tasks (with Hamiltonians H_1 to H_8). Their quantum state evolutions are initially captured by a single representative Hamiltonian, H_{L1B1}, which forms the tree root. After T_1 VQA iterations, TreeVQA decides to split into two branches based on execution similarity. After the split, H_{L2B1} is the representative Hamiltonian for the first three tasks, and H_{L2B2} is for the remaining five. This branching continues until all tasks eventually execute independently and converge to their respective solutions. The benefit of TreeVQA lies in the shared executions across all internal tree levels prior to the leaf nodes, with the greatest savings occurring near the root. Further details that justify this approach are presented in Section. 4 and beyond.

The primary contributions and insights are:

1. We make the novel observation that *substantial reductions in quantum resource costs* are possible by exploiting execution commonalities across multiple related VQA tasks within a single quantum application, an opportunity overlooked in prior work.
2. We introduce *TreeVQA*, a hierarchical execution frame-

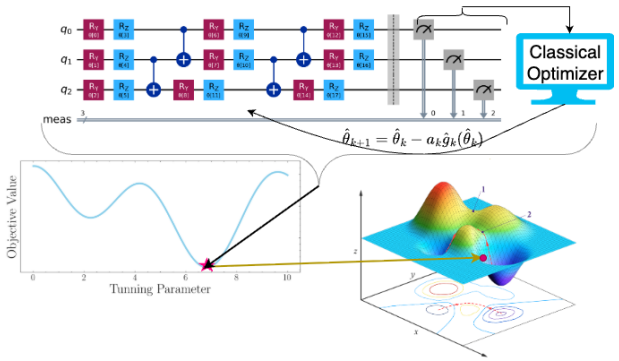

Figure 3. VQA: a hybrid algorithm alternating between classical optimization and quantum execution. Bottom right: Potential Energy Surface for a chemical reaction [43].

work that clusters VQA tasks for joint execution and adaptively branches into independent instances only as tasks diverge. This design delivers the best of both worlds: *dramatic quantum resource savings* through shared execution, while *preserving accuracy* through adaptive branching.

3. TreeVQA is designed as a *plug-and-play wrapper* that seamlessly integrates with existing VQA algorithms and optimizers, requiring minimal tuning or structural changes. We demonstrate broad applicability through results on VQE and QAOA, using SPSA and COBYLA optimizers.
4. Our evaluations show that TreeVQA reduces execution shot counts by *25.9× on average*, and by *over 100× in large-scale applications and diverse settings*, all while meeting the same fidelity targets as baseline VQA.
5. Crucially, we show that TreeVQA's benefits *amplify with problem size and precision demands*, making TreeVQA increasingly impactful as quantum applications and hardware scale toward practical use.

2 Background and Motivation

2.1 Variational Quantum Algorithms

Variational quantum algorithms (VQAs) exhibit inherent error resilience due to their hybrid structure, alternating quantum circuits with noise-aware classical optimizers [42, 48]. Prominent VQA applications include the Variational Quantum Eigensolver (VQE) [48], the Quantum Approximate Optimization Algorithm (QAOA) [13], and Quantum Machine Learning (QML) [4]. This work focuses on VQE and QAOA. VQE is widely applied in molecular chemistry, condensed matter physics, and quantum many-body systems. QAOA, on the other hand, is tailored for approximate solutions to combinatorial optimization problems, particularly Quadratic Unconstrained Binary Optimization (QUBO).

Each VQA iteration uses a parameterized quantum circuit, or *ansatz*, which defines the accessible state space and shapes the optimization. The ansatz typically consists of single-qubit rotations and fixed two-qubit gates. A classical optimizer [37, 59] iteratively updates the parameters until the objective function converges. This is shown at the top and bottom-left of Fig.3. The target problem is encoded as a Hamiltonian, represented as a linear combination of Pauli operators. The goal is to estimate its ground state energy, i.e., the lowest eigenvalue [42]. In each iteration, the objective is computed by measuring the Hamiltonian expectation using the current ansatz across appropriate measurement bases.

2.2 The Cost of Executing a Single VQA Task

VQAs typically demand fewer qubits and shallower circuits compared to many quantum algorithms. However, this comes at the expense of an exceptionally large number of total execution shots to iteratively optimize the ansatz employed to prepare the target quantum state.

[26] highlights that even relatively simple VQE problems in molecular chemistry, such as estimating the ground state energy for NH_3 (12 qubits), can require up to 5.5×10^8 execution shots. This translates to approximately 564 hours of runtime on typical cloud-based quantum devices, incurring monetary expenses on the order of $5868k, exorbitant costs for fairly modest quantum problems. Scaling to more practically useful applications, [64] estimates that a single iteration of 52-qubit VQE on the Cr_2 molecule (often considered at the threshold of quantum advantage) could demand 24 years of runtime without parallelization. Alternatively, achieving a runtime of 3ms would require 280×10^9 qubits under perfect parallelization and ideal overhead assumptions.

The aforementioned estimates exclude the substantial costs associated with error mitigation techniques such as Probabilistic Error Cancellation (PEC) [3] and Zero Noise Extrapolation (ZNE) [18], although these methods are not exclusive to VQAs. In a VQA-specific context, [9] reports that measurement error mitigation [11] can amplify the shot count of a 100-qubit VQA task by approximately 1000x. These exorbitant overheads underscore the pressing need for innovative strategies to mitigate resource consumption.

To quantitatively assess VQE cost, we focus on shot count (execution number of quantum circuits). The total number of shots depends on the number of Pauli Terms of the decomposed Hamiltonian, the convergence speed of the classical optimizer, and the statistical accuracy desired. Given a Hamiltonian expressed as a weighted sum of Pauli terms: $H = \sum_{j=1}^{M} c_j P_j, \quad c_j \in \mathbb{R}, \quad P_j \in \{I, X, Y, Z\}^{\otimes n}$, the expectation value $\langle H \rangle$ is estimated via measurements. To achieve target accuracy ϵ, the required number of shots per evaluation is approximately: $N_{\text{per-eval}} \approx \frac{\left(\sum_{j=1}^{M} |c_j|\right)^2}{\epsilon^2}$. For chemical problems, where the Hamiltonian norm is 10^2–10^3 and energy accuracy target is 10^{-3}–10^{-4} Hartree, this yields 10^6–10^8 shots per evaluation. Note that the shot count reflects the cost of a single evaluation. The overall shot cost for the full optimization process, with T iterations, is given by: $N_{\text{overall}} = T \times N_{\text{evals-per-iter}} \times N_{\text{per-eval}}$. The iteration num-

ber T is affected by convergence speed, while the number of evaluations per iteration is affected by the classical optimizer.

2.3 Energy Landscapes from Multiple VQA Tasks

A primary application of VQAs (specifically VQE) is estimating a molecule's ground state energy - a task that scales exponentially on classical computers [21] and enables predictions of molecular stability, reactivity, and dynamics. Comparing ground state energies across different molecular configurations is especially critical as it reveals insights about reaction pathways and outcomes. These relationships are captured by the potential energy surface (PES), which maps a molecule's energy as a function of atomic positions. The PES (shown to the bottom right of Fig.3) highlights key features such as minima (stable states), maxima (transition states), and saddle points. High-accuracy ground state estimates improve the PES resolution, enabling more reliable predictions of molecular stability, vibrational spectra, and dynamic behavior [23].

Constructing such landscapes with VQAs requires solving multiple VQA tasks, since each molecular configuration corresponds to a separate Hamiltonian with a unique ground state energy. A finer landscape that faithfully captures small variations between nearby points demands both higher precision in each task and a larger number of tasks overall. In the PES shown to the bottom right of Fig.3, a single point on the energy surface corresponds to a VQA task, and many 1000s of tasks are required to construct the entire landscape.

The concept of constructing detailed energy landscapes via VQA tasks extends to other domains. In condensed matter physics, VQAs can model the electronic ground state to investigate material properties and phase transitions, with energy surfaces shaped by atomic configurations and interaction strengths [46]. In protein folding, energy landscapes determine folding pathways through numerous local minima and transition states [53]. VQAs can help map these complex landscapes. Analogous to VQE in physical systems, constructing 'energy' landscapes with QAOA is also highly relevant and involves solving multiple related optimization tasks, each typically corresponding to a graph instance with specific edge weights. These optimization problems can correspond to a wide variety of real-world use cases, such as in weather forecasting, traffic management, supply chain, smart grids (which we will detail later), and more.

3 Observation: VQA Task Similarity

The central insight motivating our execution model is the inherent similarity among the many VQA tasks that collectively define an application's energy landscape.

The theoretical foundation of this approach is rooted in the **adiabatic theorem** [5], which establishes that for a system described by a slowly varying Hamiltonian, the ground state wavefunction evolves continuously. Specifically, if a Hamiltonian H_1 is adiabatically transformed into a nearby Hamiltonian H_2 in the system's parameter space, the corresponding ground state wavefunction ψ_1 will evolve smoothly into ψ_2. This adiabatic evolution is contingent upon the persistence of a non-zero energy gap ($\Delta E > 0$) between the ground and first excited states. If this gap were to close, the system would approach a quantum phase transition, permitting non-adiabatic transitions and a potentially discontinuous change in the ground state's character. This principle is illustrated in Figure 4a, which presents the ground state wavefunction of the H_2 molecule at five bond lengths. In this representation, the state is expanded in a basis corresponding to the occupation of molecular orbitals by the two covalent electrons. The figure demonstrates that the coefficients of the primary basis states, such as the covalent configurations $|01\rangle$ and $|10\rangle$, vary very gradually and in close relation to the changes in the internuclear distance.

To further demonstrate that systems with similar configurations (e.g., bond lengths for molecules) exhibit correlated behavior, we construct Fig. 4b, which quantitatively shows the overlap between ground states of VQE tasks at different bond lengths for LiH dissociation. The heatmap places bond length along both axes, with each point (x, y) denoting the overlap between ground states at those lengths. Bright yellow regions correspond to high overlap (evident at small bond length differences), while overlap diminishes at extreme bond lengths. Complementing this, Fig.4c illustrates that such similarity can be inferred directly from the task Hamiltonians. Specifically, we compare the task Hamiltonians via their non-trivial Pauli coefficients using the ℓ_1 norm, and compute affinities through a Gaussian kernel (details in Section 5.2.4). This coefficient-based approach yields consistent insights, confirming that neighboring tasks share structural and spectral similarity.

These observations have direct implications for TreeVQA. Since the VQA's quantum state is prepared by a parameterized unitary transformation, the continuity of the ground state wavefunction implies that the optimal set of variational parameters is also a continuous function of the system's Hamiltonian. The TreeVQA algorithm is designed to capitalize on this insight. It leverages the expected continuity of optimal parameters to guide and accelerate the optimization process for a family of related Hamiltonians. Furthermore, TreeVQA incorporates a dynamic monitoring mechanism capable of detecting signatures of a quantum phase transition, such as abrupt changes in the optimization landscape, allowing it to adapt its strategy and effectively navigate these critical points. Finally, although these ideas are motivated here in the context of molecular chemistry, the underlying principle generalizes across domains.

4 TreeVQA: Tree-based VQA Execution

TreeVQA is a hierarchical framework designed to efficiently handle applications involving multiple variational quantum

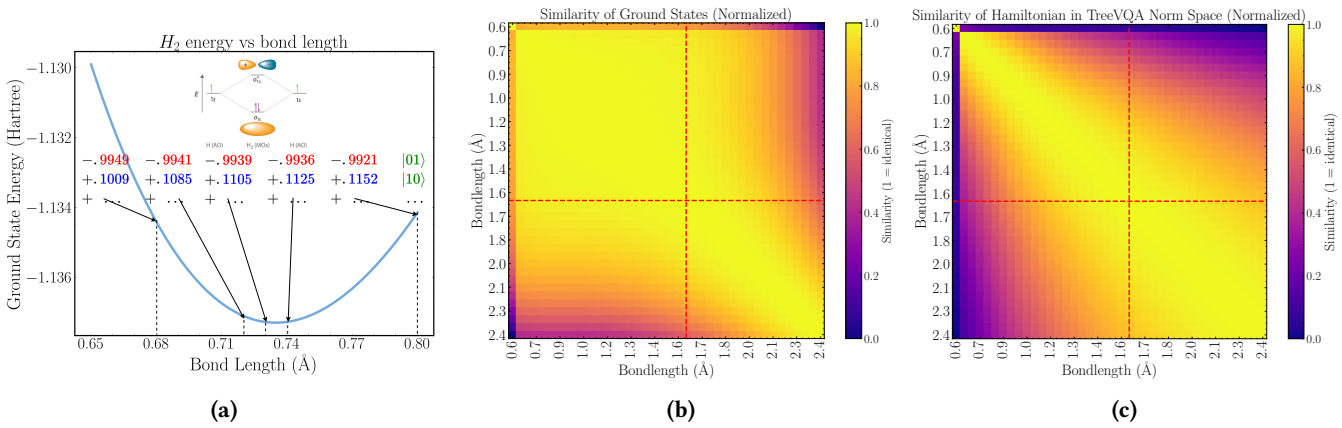

Figure 4. (a) Ground states of H_2 and its Molecular Orbital schematic [39] (b) Similarity of ground states of LiH at different bond lengths. (c) Hamiltonian similarity across the VQA tasks, computed through TreeVQA similarity metric (Section 5.2.4).

algorithm (VQA) tasks with underlying similarity. By clustering related tasks, TreeVQA adaptively optimizes shared circuit parameters across groups, thereby reducing the total number of quantum measurements. As optimization proceeds, clusters are recursively subdivided, enabling increasingly fine-grained adaptation to each Hamiltonian. Fig. 2b illustrates this process for an application with eight VQA tasks, each associated with a distinct Hamiltonian (H_1 through H_8). For example, these Hamiltonians could represent a molecule evaluated at eight different geometries (e.g., bond lengths), where each H_i corresponds to the molecular Hamiltonian at a specific geometry. The objective is then to find the ground state at each geometry. TreeVQA execution is detailed below:

1. To begin, all VQA tasks are initially grouped into a single cluster, as they may share a suitable reference quantum state (note: this is not required in TreeVQA - one could also start with multiple clusters if desired). Each cluster corresponds to a single "mixed" Hamiltonian that represents all task Hamiltonians within it (see Section 5.2.1 for details). In Figure 2b, this is shown in blue as H_{L1B1} (Level1, Branch1). Variational optimization is then performed on an ansatz, with its parameters updated based on the expectation value of the mixed Hamiltonian. This single-cluster execution continues until branching events occur, as described next.
2. After a warmup phase, the optimization of each VQA cluster is monitored using a sliding window method that evaluates slopes (see Sections 5.2.2). When a split condition is triggered, signaling either stagnation in optimization or divergence among Hamiltonians within the cluster, the cluster is divided. The split is guided by precomputed task similarity information (see Section 5.2.3). In Figure 2b, after T_1 iterations of single-cluster execution, the process branches into two clusters, shown in green.
3. The splitting process employs similarity-based clustering (see Section 5.2.5) to identify natural groupings of Hamiltonians. In the example, the first three tasks exhibit similar characteristics and are grouped into one cluster with mixed Hamiltonian H_{L2B1} (Level 2, Branch 1), while the remaining five tasks form a second cluster with mixed Hamiltonian H_{L2B2} (Level 2, Branch 2). Each new cluster inherits the parameters of its parent, enabling warm-started optimization.
4. This hierarchical splitting process continues recursively; for instance, the second branch may further divide into two level-three branches (yellow). Each split occurs only when the specified conditions are met, ensuring efficient utilization of quantum resources. The procedure terminates either when the total shot budget is depleted or when convergence is achieved (see Section 5.1). Additionally, post-processing steps are applied to refine the results (not shown in the figure, see Section 5.3).

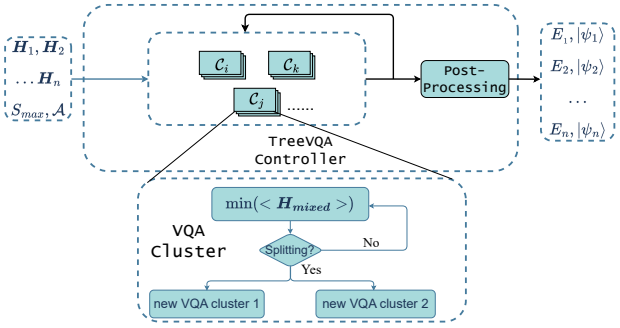

Figure 5. TreeVQA: A central controller and VQA clusters

5 TreeVQA Framework Architecture

The TreeVQA framework is designed as a central controller to orchestrate the execution and communication among the VQA clusters. We discuss these next.

5.1 TreeVQA Central Controller

The Central Controller orchestrates the overall execution of the algorithm, including cluster creation, splitting logic, resource coordination, and enforcement of the global shot budget, as illustrated in Figure 5. The controller workflow is detailed in Algorithm 1.

For a given application, TreeVQA receives the set of task Hamiltonians and their initial states (e.g., Hartree–Fock, CAFQA, or Red-QAOA [63]; see section 8 for details). Based on the number of unique initial states, TreeVQA initializes a corresponding number of VQA clusters, each containing Hamiltonians sharing the same initial state. The controller manages the entire process: each VQA cluster optimizes its assigned Hamiltonians and may split into child clusters as necessary (see subsection 5.2), with child clusters inheriting parameters from their parent. After all iterations, a final post-processing step evaluates each Hamiltonian across all cluster states to select the optimal solution (see subsection 5.3).

Algorithm 1 TreeVQA Global Controller

Input: Hamiltonians $\{H_1, \ldots, H_N\}$, total shot budget $S_{\max}$, ansatz $\mathcal{A}$
Initialize root cluster $C_0 \leftarrow \{H_1, \ldots, H_N\}$
$\mathbb{C} \leftarrow \{C_0\}, \quad S_{\text{total}} \leftarrow 0$
while $S_{\text{total}} < S_{\max}$ **do**
  for each cluster $C \in \mathbb{C}$ **do**
    VQA-Cluster-Step(C) ▷ See Algorithm 2
    $S_{\text{total}} \leftarrow S_{\text{total}} + S_C$ ▷ Cluster shot usage
  end for
  Update $\mathbb{C}$ by replacing split clusters with their children
end while
$\mathbb{C}_{\text{final}} \leftarrow \mathbb{C}$ ▷ Final cluster set
for each Hamiltonian H_i **do**
  for each cluster $C \in \mathbb{C}_{\text{final}}$ **do**
    $E_i^C \leftarrow \langle\psi(\theta_C)|H_i|\psi(\theta_C)\rangle$ ▷ Post-processing evaluation
  end for
  $C_{\text{best}}(H_i) \leftarrow \text{argmin}_{C \in \mathbb{C}_{\text{final}}} E_i^C$ ▷ Best state for each Hamiltonian
end for
Return: For each H_i, the best energy $E_i^{C_{\text{best}}}$ and corresponding state $\psi(\theta_{C_{\text{best}}(H_i)})$

5.2 VQA Cluster Functionality

A *VQA Cluster* is a fundamental computational unit in the TreeVQA framework (see zoomed region in Figure 5), responsible for jointly optimizing a shared parameterized quantum state $|\psi(\theta)\rangle$ over a subset of Hamiltonians $\{H_1, H_2, \ldots, H_N\}$. By aggregating the Pauli terms from its assigned Hamiltonians, the cluster constructs a mixed Hamiltonian and performs collective parameter optimization to approximate the ground states of all members, leveraging their similarity. If the optimization trajectories of certain Hamiltonians diverge significantly (quantified by a similarity metric) the cluster adaptively splits into child clusters to maintain optimization efficiency and accuracy. We detail the functionality of the VQA Cluster and the design philosophy behind it in the following subsections.

Algorithm 2 VQA-Cluster-Step

Compute the mixed Hamiltonian H_{mixed}
Compute similarity matrix S_C
Initialize parameters θ_C (inherited from parent)
while true **do**
  Optimize θ_C over the mixed Hamiltonian.
  Track loss values for the mixed and individual Hamiltonians.
  if warmup period is complete **then**
    Compute $\text{slope}_{\text{mixed}}$ via linear regression on mixed loss
    Compute slope_i for each $H_i \in \mathbb{H}_C$
  end if
  if $\text{slope}_{\text{mixed}} < \epsilon_{\text{split}}$ **or** any $\text{slope}_i > 0$ **then**
    Split C into C_1, C_2 via spectral clustering on S_C
    Inherit parameters: $\theta_{C_1} = \theta_{C_2} = \theta_C$
    Mark C as retired
    return C_1, C_2
  end if
end while
Record shot usage S_C

5.2.1 Cluster Mixed Hamiltonian Construction.

When the target application's Hamiltonians vary over its tasks, small but nonzero integrals can emerge or vanish [62], resulting in new or missing Pauli terms in its qubit Hamiltonian. To handle multiple Hamiltonians consistently, the VQA cluster first identifies the superset of all Pauli terms: $\mathcal{P} = \bigcup_{i=1}^{N} \{P_j \mid P_j \in H_i\}, \quad |\mathcal{P}| = M$. Then, for each Hamiltonian H_i, it pads any missing terms in $\mathcal{P}$ with zero. Since many of these extra terms arise from integrals that are either zero or negligibly small, the padding is typically minimal, and any new coefficients are close to zero. Thus, while padding ensures consistency across VQA tasks, it does not significantly increase the Hamiltonian's complexity or alter its underlying characteristics.

The cluster's mixed Hamiltonian (which represents the set of Hamiltonians this cluster will handle) is given by: $H_{\text{mixed}} = \frac{1}{N}\sum_{i=1}^{N} H_i^{\text{padded}} = \sum_{k=1}^{M}\left(\frac{1}{N}\sum_{i=1}^{N} c_{ik}\right) P_k$. This construction yields a Hermitian operator according to the linearity of quantum mechanics, incurs minimal computational cost, and provides a representative optimization landscape for all cluster members. The corresponding objective function is: $\mathcal{L}(\theta) = \langle\psi(\theta)|H_{\text{mixed}}|\psi(\theta)\rangle = \frac{1}{N}\sum_{i=1}^{N}\langle\psi(\theta)|H_i^{\text{padded}}|\psi(\theta)\rangle$.

5.2.2 Iterative VQA Optimization and Sliding Window Slope Monitoring. After the preliminary step, VQA optimization is performed on each cluster's mixed Hamiltonian, iteratively updating the cluster's ansatz parameters θ using the selected VQA optimizer. In this work, we focus on the Simultaneous Perturbation Stochastic Approximation (SPSA) optimizer [59], though the framework can be generalized to other optimizers. The parameter update is given by: $\theta_{t+1} = \theta_t - \eta_t \frac{\mathcal{L}(\theta_t+\Delta_t)-\mathcal{L}(\theta_t-\Delta_t)}{2\Delta_t}$, where Δ_t is a random perturbation vector and η_t is the learning rate.

After T_{warmup} steps, allowing the optimizer to freely explore the parameter space without splitting constraints, we compute a running slope over a window of W loss values: $\text{slope}_t = \text{LinearRegression}\left(\{\mathcal{L}(\theta_{t-W+1}), \ldots, \mathcal{L}(\theta_t)\}\right)$. For each Hamiltonian H_i in the cluster, we also track $\text{slope}_{i,t}$ after each iteration in the same way, where $\mathcal{L}_i(\theta) = \langle\psi(\theta)|H_i|\psi(\theta)\rangle$. Notably, this individual tracking incurs only a linear classical computational cost, not a quantum execution cost. This tracking is used in the next subsection.

5.2.3 Cluster Splitting. The tracking described above allows for cluster splitting when appropriate, ensuring that each VQA task reaches its own target. We trigger the splitting if either of the following conditions is met: $|\text{slope}_t| < \epsilon_{\text{split}}$ or $\exists\, i$, such that $\text{slope}_{i,t} > 0$ These conditions capture both a stalled overall optimization and an unproductive path for any individual Hamiltonian within the cluster.

If splitting is not triggered, the optimization will continue until retirement. However, when splitting occurs, it indicates that the cluster has determined that some Hamiltonians are too distant from others to be optimized together. As a result, the cluster will be divided into several new clusters. The key challenge is to identify which Hamiltonians are "far" from the others and which are "close," i.e., to quantify distances between Hamiltonians. Moreover, the distance should capture both gradient direction and ground-state energy shift while remaining efficient to compute; we discuss this next.

5.2.4 Hamiltonian Similarity: Hilbert Space Distance and Gradient Alignment. To quantify the similarity between Hamiltonians H_i and H_j, we compute the ℓ_1 norm between their padded coefficient vectors:

$\mathbf{c}_i = (c_{i1}, \ldots, c_{iM}), \quad d(H_i, H_j) = \|\mathbf{c}_i - \mathbf{c}_j\|_1 = \sum_{k=1}^{M} |c_{ik} - c_{jk}|$. This metric is computationally efficient and serves as a practical proxy for Hilbert space proximity, as the operator norm difference between two Hamiltonians is bounded by their ℓ_1 coefficient distance:

$\|H_i - H_j\|_{\text{op}} = \left\|\sum_{k=1}^{M}(c_{ik} - c_{jk})P_k\right\|_{\text{op}} \leq d(H_i, H_j)$. According to perturbation theory [55], if $\|H_i - H_j\|_{\text{op}} \ll \Delta$ (where Δ is the spectral gap), the ground-state energies and eigenstates of H_i and H_j remain close. Thus, a small ℓ_1 distance indicates similar ground-state properties and optimization landscapes.

To facilitate smooth and normalized pairwise similarity, we construct a Gaussian (RBF) kernel: $S_{ij} = \exp\left(-\frac{d(H_i,H_j)^2}{2\sigma^2}\right)$, where σ is set to the median pairwise distance in the dataset. The resulting similarity matrix $S \in \mathbb{R}^{N\times N}$ guides cluster splitting. Beyond static similarity, the ℓ_1 distance also reflects the alignment of gradient directions during VQA optimization. The gradient of the loss function $\langle\psi(\theta)|H_i|\psi(\theta)\rangle$ with respect to parameters θ is determined by the dominant Pauli coefficients in H_i. Hamiltonians with similar coefficient vectors yield aligned gradients, enabling effective joint optimization within a cluster. Conversely, a Hamiltonian with a distinct coefficient vector will have a divergent gradient direction, impeding convergence if grouped with dissimilar tasks. The similarity matrix S thus not only encodes Hilbert space proximity but also captures the coherence of optimization trajectories. When the similarity matrix is available, spectral clustering [67] is employed to partition the cluster when the splitting condition is met, as shown below.

5.2.5 Cluster Assignment via Spectral Clustering. Upon triggering a cluster split, we apply spectral clustering to the similarity matrix S to partition the Hamiltonians into homogeneous subgroups. Spectral clustering constructs the normalized Laplacian of S, computes its leading eigenvectors, and performs k-means clustering in this reduced space to assign each Hamiltonian to one of two child clusters [67]. This approach efficiently identifies natural divisions in the similarity structure, ensuring that Hamiltonians with greater similarity are grouped together. Each child cluster inherits the parent's parameter values, enabling warm-started optimization and continuity in the variational process.

5.3 Post-Processing

After reaching the shot budget limit ($S_{\text{total}} > S_{\max}$), TreeVQA performs post-processing to ensure optimal results for each Hamiltonian. At this stage, we have a set of final clusters $\mathbb{C}_{\text{final}}$, each with its own optimized quantum state $|\psi(\theta_C)\rangle$ as shown in Algorithm 1. For each individual Hamiltonian H_i, we evaluate its expectation value on *all* optimized states from the final clusters. We then select the best state for each Hamiltonian based on these expectation values. The post-processing approach is computationally efficient because each cluster already logs the expectation values of individual Pauli terms during optimization. Evaluating a Hamiltonian on states from different clusters is essentially a classical recombination of stored values with different coefficients.

6 Tailoring for QAOA

TreeVQA is broadly suited for all VQAs, but its implementation is more directly suited to VQE. In this section, we discuss tailoring it for QAOA.

The QAOA ansatz consists of a circuit with p alternating parameterized layers acting on an initial state $|\Psi_0\rangle$. Each layer ℓ applies the phasing operator $e^{-i\gamma_\ell C}$, adding a complex phase to each bitstring x based on its cost $c(x)$ from

the QUBO and a real-valued parameter γ_ℓ. This phasing step is applicable to any QUBO [19]. The mixing layer, $e^{-i\beta_\ell B}$, combines quantum state amplitudes, with the coefficient depending on the Hamming distance between states and a real-valued parameter β_ℓ. Specifically, $B = \sum_i X_i$, so $e^{-i\beta_\ell B}$ can be implemented as single-qubit R_x rotations on each qubit. The goal is to optimize the $2p$ parameters $(\vec{\gamma}, \vec{\beta})$ to maximize quantum amplitudes for optimal or near-optimal bitstrings x. The performance of QAOA depends on the choice of parameters $(\vec{\gamma}, \vec{\beta})$, and optimizing these parameters can be highly non-trivial [2, 8, 14, 15, 27, 28, 40, 41, 60, 69]. A common approach is variational optimization, where a quantum computer estimates the expectation value of the quantum state induced by QAOA with parameters $(\vec{\gamma}, \vec{\beta})$ with respect to the cost operator C. A classical computer then updates $(\vec{\gamma}, \vec{\beta})$ using this estimate, along with previous ones, and the quantum-classical loop continues until a convergence criterion is met.

The standard QAOA ansatz is problem-specific and minimally parameterized, typically using only $2p$ parameters. To implement TreeVQA effectively, we adopt a circuit ansatz that generalizes across problem instances, enabling the representation of problems with shared structure. For finer control in splitting during TreeVQA, we increase the number of classical parameters per QAOA layer. We achieve this by implementing the multi-angle QAOA (ma-QAOA) [29], which enhances the expressivity of the traditional structure by assigning individual parameters to each term in the cost and mixer operators, instead of using a single angle for each. Specifically: $U(C, \vec{\gamma}_\ell) = e^{-i\sum_{a=1}^{m} C_a \gamma_{\ell,a}} = \prod_{a=1}^{m} e^{-iC_a\gamma_{\ell,a}}$, and $U(B, \vec{\beta}_\ell) = e^{-i\sum_{b=1}^{n} B_b\beta_{\ell,b}} = \prod_{b=1}^{n} e^{-iB_b\beta_{\ell,b}}$. Here, m is the number of clauses and n is the number of qubits. The total number of parameters becomes $(m+n)p$. The standard QAOA is a special case of ma-QAOA, where all parameters within a cost or mixer layer are identical. ma-QAOA expands the parameter space, allowing TreeVQA to explore more complex solution landscapes and improve optimization efficiency.

7 Methodology

7.1 Benchmarks

Table 1. Chemistry Benchmarks

	H_2	LiH	BeH_2	HF	$C_2H_2^*$
# Ham. terms	15	496	810	631	5945
Qubit numbers	4	12	14	12	28
Bond range (Å)	.74-.83	1.4-1.7	1.2-1.47	.83-1.1	1.15-1.25
Eq. bond (Å)	.741	1.595	1.333	.917	1.2

Chemistry Benchmarks: We select H_2, BeH_2, LiH, HF, and C_2H_2 as primary chemical benchmarks. Using Qiskit's Jordan–Wigner mapper [65], we convert the fermionic Hamiltonian in the STO-3G basis to a qubit Hamiltonian, with the resulting terms and qubit counts shown in Table 1. All molecular spin orbitals are left open [12], enabling TreeVQA to distinguish configurations near equilibrium. We initialize both TreeVQA and the baseline to start with the Hartree-Fock state unless specified otherwise. Each benchmark includes 10 bond-length instances spaced by 0.03 Å. For H_2, we use the UCCSD ansatz with 5 instances due to its smaller size. TreeVQA is also applicable to a broader range of molecule geometries.

Physics Benchmarks: We evaluate two spin-$\frac{1}{2}$ models: (1) The Heisenberg XXZ chain with Hamiltonian $H_{\mathrm{XXZ}} = J\sum_{i=1}^{N-1}(\sigma_x^i\sigma_x^{i+1}+\sigma_y^i\sigma_y^{i+1}+\Delta\sigma_z^i\sigma_z^{i+1})$, where Δ is the anisotropy parameter that drives transitions between gapless ($|\Delta| < 1$) and gapped phases, with a BKT transition[61] at $\Delta = 1$; and (2) The transverse Ising model with $H_{\mathrm{TI}} = -J\sum_{i=1}^{N-1}\sigma_z^i\sigma_z^{i+1} - h\sum_{i=1}^{N}\sigma_x^i$, which exhibits a quantum phase transition at $h = J$ between ordered and disordered phases[17]. For both models, we set $J = 1$ and use Qiskit's LinearMapper[50] for spin-to-qubit mapping.

QAOA Benchmark: The Maximum Cut (MaxCut) is a well-studied NP-hard problem in graph theory, widely used as a benchmark for QAOA. It has applications in statistical physics and circuit design, and involves partitioning the vertices of a graph into two complementary subsets to maximize the sum of edge weights crossing the cut. The cost Hamiltonian for a graph G with edge set $E(G)$ is $H_{\mathrm{C}} = \sum_{(i,j)\in E(G)} \frac{1}{2}w_{ij}\left(I - Z_iZ_j\right)$, where w_{ij} are edge weights. We benchmark TreeVQA on MaxCut as it enables exploration of diverse graph structures under uniform optimization settings. In particular, we evaluate graph instances derived from the IEEE 14-bus test system [66], a canonical dataset in power systems. Solving MaxCut on these graphs identifies optimal partitions that maximize power transfer while minimizing inter-partition dependencies; this is key for system reliability and distributed energy coordination in smart grids [33]. In these graphs, buses correspond to vertices and transmission lines/transformers to edges, yielding a weighted graph well-suited for QAOA. We construct isomorphic graph representations where nodes denote buses and weighted edges capture line capacities influenced by load conditions. By scaling load values, we vary edge weights to simulate operating regimes from light to heavy demand.

7.2 Evaluation Metrics

We define each benchmark instance, such as molecular configurations at different bond lengths or physical models at varying parameters, as a *task*. For each task i, with E_{gs_i} denoting the ground-state energy, the error is defined as $\epsilon_i = \frac{E_{gs_i} - E_i}{E_{gs_i}}$, with fidelity $F_i = 1 - \epsilon_i$. Since TreeVQA collectively solves multiple tasks for an application, we aggregate error and fidelity across the N subproblems. We define the fidelity threshold T: $\forall F_i, i \in \{1, 2, \ldots, N\}, \quad F_i \geq T$.

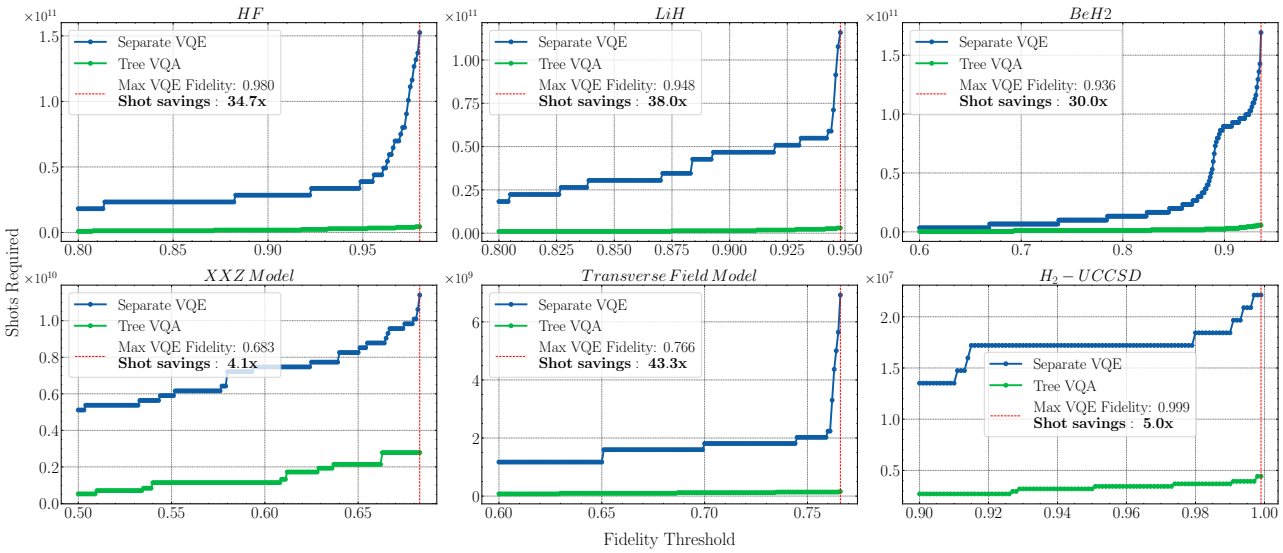

Figure 6. Shot reduction for TreeVQA compared to the baseline, for a fixed fidelity target across benchmarks.

7.3 Baseline and Evaluation Parameters

The baseline is conventional VQA, where each task within an application is executed independently with an equal allocation of shots. Unless otherwise specified, we employ the SPSA optimizer [59] with a mini-batch size ($N_{\text{evals-per-iter}}$) of 2. For rigorous and consistent comparison, we set $N_{\text{per-eval}} = 4096*$ (number of Pauli terms) for all VQA tasks in both the baseline and TreeVQA. Thus, each Pauli term is sampled 4096 times per evaluation, ensuring statistical accuracy in the measurement outcomes (see subsection 2.2). As detailed in Section 2.2, this is a conservative estimate; for larger Hamiltonians, TreeVQA can yield even greater shot reductions. The total shot count is given by $N_{\text{overall}} = (\#\text{ iterations}) \times 2 \times 4096 \times$ (number of Pauli terms). While $N_{\text{per-eval}}$ could be further reduced by grouping Pauli terms into Qubit-Wise Commuting (QWC) sets, this constant-factor improvement does not affect the relative shot reduction ratio, which is our primary metric. Benchmark-specific shot budgets are chosen to ensure TreeVQA reaches convergence. Iteration counts typically range from 16,000 to 30,000, except for the simpler H_2 system, which converges within 300 iterations.

7.4 Simulation Framework

TreeVQA is evaluated using Qiskit's `AerSimulator` [32]. For noiseless simulations, we utilize the `StatevectorSimulator` to obtain exact quantum state distributions. For realistic noise studies (see Section 8.7), we employ Qiskit's density matrix simulator with device-calibrated noise models, incorporating both discrete Pauli errors (from reset, measurement, and gate operations) and continuous decoherence processes (T1, T2). To enable scalable simulation of large systems, we leverage the PauliPropagation method [54], which provides efficient, low-error simulation beyond the capabilities of conventional approaches, enabling studies of systems such as C_2H_2 and 25-site Ising chains. For all noiseless experiments, we use the Hardware Efficient Ansatz [1] with two layers of circular entanglement; in noisy settings, we increase to five layers to amplify the effects of noise on fidelity.

8 Evaluation

8.1 VQA Shot Reduction for a Fidelity Target

The goal of this evaluation is to estimate the VQE shot savings achieved by TreeVQA over the baseline, which executes each VQE task independently, for different fidelity thresholds. Results are shown in Figure 6, comparing the total shots required by TreeVQA and the baseline across all tasks.

Across all six VQE benchmarks, TreeVQA consistently yields substantial shot savings. For instance, in the *HF* molecule case, the baseline VQE attains about 98% fidelity across tasks for $\sim 1.5 \times 10^{11}$ shots, whereas TreeVQA achieves the same fidelity with only 4×10^9 shots — a 34.7× reduction. As the target fidelity approaches 1.00, TreeVQA's shot count rises because clustered tasks are eventually separated into distinct VQE instances to reach high-accuracy solutions, each requiring multiple iterations to converge. Importantly, the baseline approach would also consume these shots and would have already expended a substantial number to reach this fidelity, often making simulation impractical. Thus, TreeVQA remains more resource-efficient across the fidelity range.

Similar trends appear across other molecular and physical systems. Shot savings typically range from 30× to 40×, though some cases differ. In the H_2 UCCSD benchmark, the small problem size allows both methods to converge in fewer iterations, yielding smaller savings. In the XXZ model, slow convergence causes TreeVQA to split tasks aggressively, slightly reducing its relative advantage. These

cases suggest that further refinements, such as more adaptive threshold tuning, could improve TreeVQA's performance on more challenging optimization landscapes.

An interesting finding is that TreeVQA often surpasses the nominal shot-saving bound set by subproblem size. This arises from its effect on the SPSA learning rate, $\eta_k = \frac{a}{(A+k+1)^\alpha}$, where the superposition of Hamiltonians increases landscape curvature, driving larger a and thus steeper gradients. The higher learning rates speed convergence and persist even when baselines are tuned, highlighting TreeVQA's efficiency.

8.2 Fidelity Benefit for a Fixed Shot Budget

Another way to interpret TreeVQA's benefit is through fidelity improvements under a fixed shot budget. As shown in Figure 7, TreeVQA consistently achieves higher fidelity compared to the baseline.

Additionally, TreeVQA leads to lower fidelity variance in fidelity across different tasks (e.g., varying bond lengths or magnetic field strengths). This is because the tasks in TreeVQA are solved in a coordinated fashion, guided by shared information within clusters. In contrast, baseline VQEs—executed independently—often exhibit large disparities in fidelity under a uniform shot budget, since the optimal allocation for each task is unknown in advance.

8.3 Benefits at Varying Task Precision

Figure 8 shows that TreeVQA's shot savings increase with higher precision requirements (smaller step sizes) over a fixed bond length range. As precision increases, the number of tasks grows (3, 5, 7, 10, and 30). Due to computational constraints, results for the finest step size (0.001) are extrapolated, indicated by shaded bars. Shot savings rise from 5–10× at coarse precision to 80–100× at finer precision, exceeding 250× at the highest precision. This trend is intuitive: higher precision creates more subproblems with highly similar Hamiltonians, which can be effectively co-optimized within a single VQA cluster, maximizing quantum resource efficiency. Since practical applications often require high-precision energy estimates, TreeVQA is expected to provide substantial benefits in real-world scenarios.

8.4 Estimating Benefits for Large-Scale Problems

To evaluate TreeVQA on large-scale problems that may approach near-term quantum utility, we consider two benchmarks: C_2H_2 and a 25-site Ising model (each with ten tasks), using 28 and 50 qubits, respectively. Simulations are performed with the PauliPropagation method, truncating Pauli terms with frequencies above 10 [54]. Results are shown in Figure 9. As classical solvers cannot provide exact ground states for these systems, we allocate TreeVQA a budget of 10,000 iterations and measure the number of shots the baseline VQE requires to reach comparable energies. For some tasks, the baseline does not achieve TreeVQA's energies within the given shots (indicated with hatch lines), suggesting that the actual shot savings will be even higher. Overall, TreeVQA demonstrates substantial shot reductions over the baseline across both benchmarks. The lower benefit observed for C_2H_2 compared to the Ising model partly reflects the limited computational budget allocated to C_2H_2 (due to runtime constraints). In fact, the average energy gap between TreeVQA and the baseline is greater for C_2H_2 than for the Ising model. With additional budget, the shot savings for C_2H_2 would likely approach those observed for the Ising.

Additionally, we also evaluate the TreeVQA on the above setting under realistic noise. We use circuit-level noise model [6, 16] ($p_{single-qubit-gate} = 0.1\%$, $p_{two-qubit-gate} = 0.5\%$ and $p_{readout} = 0.05\%$).

8.5 TreeVQA integrated with good VQE Initialization

Although TreeVQA significantly reduces quantum hardware execution (shot) costs for VQAs, it is not the only approach to do so. Shot costs can also be reduced via classical initialization: if good VQA parameters are identified classically, quantum execution can start from these parameters, accelerating convergence. For VQE, the state-of-the-art classical initialization method is CAFQA [51], which finds initial parameters using a Clifford-only search strategy. While CAFQA achieves high initialization accuracy, its benefits (and those of any classical initialization) diminish as the inherent *quantum-ness* of the problem increases, since classical methods can only resolve the "easy" classical component, leaving the harder quantum portion to iterative quantum execution.

Even with accurate classical initialization, TreeVQA provides substantial additional benefits. In Fig. 10, we evaluate the LiH molecule at a bond length of $3.15-3.24$ at a precision of 0.01 Å, where CAFQA achieves 95.5% initialization accuracy. Using CAFQA parameters to initialize both baseline VQE and TreeVQA, we find that TreeVQA recovers the residual energy gap between the CAFQA initialization and the true ground state with far fewer shots. Specifically, TreeVQA recovers 30% of this gap using 7.3× (i.e. 1.8×10^{10}) fewer shots. These benefits are expected to increase for more challenging problems where classical initialization is far less effective.

8.6 Untuned TreeVQA with Alternate Optimizer

In previous sections, we demonstrated the benefits of TreeVQA with the SPSA optimizer, one of the most widely used optimizers for VQAs. To maximize performance with SPSA, we carefully tuned TreeVQA's strategies and hyperparameters to align with SPSA's functionality (see Section 9.1 and Section 9.1). However, TreeVQA is not tied to any single optimizer. In this section, we highlight its effectiveness with an alternate optimizer, COBYLA (Constrained Optimization BY Linear Approximations), when running VQE across diverse applications. COBYLA can be advantageous in noisy, non-smooth optimization landscapes since it does not rely on gradients, but it scales poorly with large parameter counts

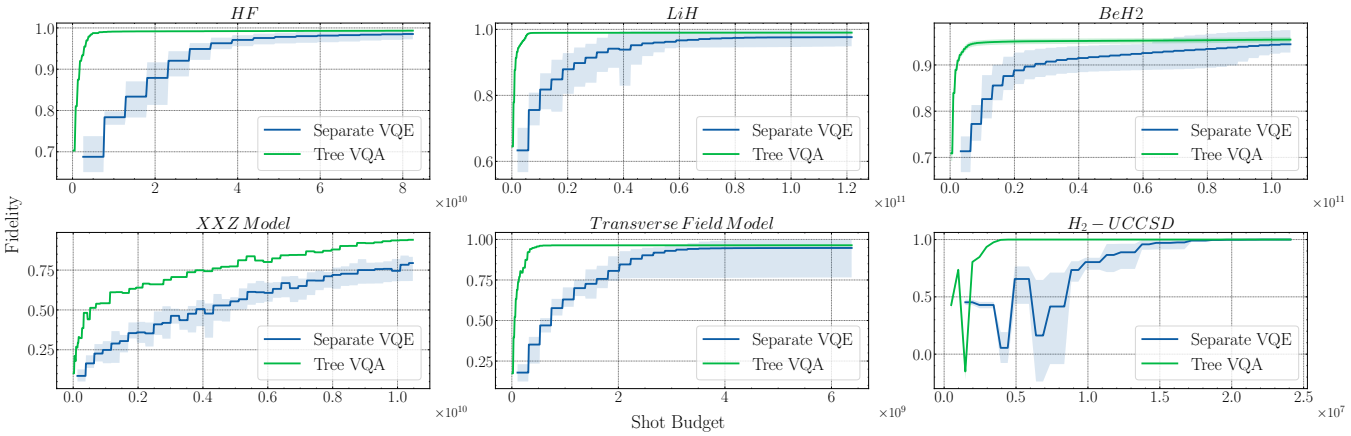

Figure 7. Fidelity gain achieved by TreeVQA compared to baseline execution for a fixed shot budget across all benchmarks

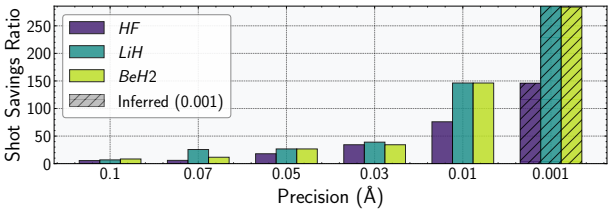

Figure 8. Shot savings at different precision levels

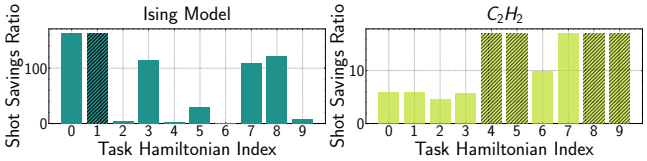

Figure 9. Shot savings on large-scale applications.

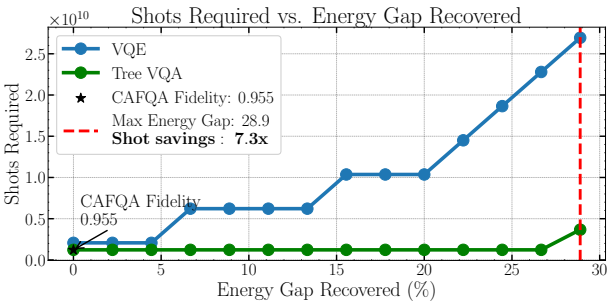

Figure 10. TreeVQA benefits for *LiH* w/ CAFQA initialization.

and can become trapped in local minima. Unlike SPSA's stochastic gradient-based updates, COBYLA relies on local linear approximations, making its optimization dynamics fundamentally different, potentially posing a challenge for TreeVQA. Thus, demonstrating TreeVQA's benefits in this setting, without any additional fine-tuning, underscores its

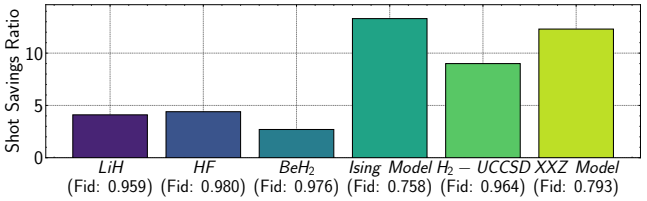

Figure 11. TreeVQA benefits for VQA w/ COBYLA optimizer.

broad applicability. Remarkably, TreeVQA still achieves 2.5×–13× shot savings over traditional VQE with COBYLA, clearly showcasing its plug-and-play utility across optimizers.

Table 2. LiH TreeVQA Noisy Simulation Results

Backend Name	Max Avg Fidelity	Shots Saving Ratio @ Max Fidelity
Hanoi	0.951	12.0×
Cairo	0.958	17.0×
Mumbai	0.910	12.1×
Kolkata	0.878	24.8×
Auckland	0.888	14.7×

8.7 Benefits in Noisy Execution Settings

All prior evaluations in this work are based on noiseless simulations. Since TreeVQA primarily modifies the VQA execution strategy rather than the hardware itself, benefits observed in noiseless settings are expected to largely carry over to real quantum devices. To validate this, we evaluate TreeVQA using noisy device simulation with *LiH* as a benchmark. To accentuate noise effects, we increase the ansatz entanglement layers from 2 to 5, as deeper circuits are both more sensitive to noise and more representative of problem-tailored ansatz, which typically exceed the depth of the Hardware Efficient Ansatz [44].

For noisy simulations, we employ Qiskit's highest opti-

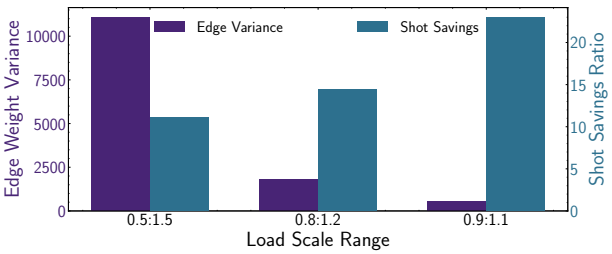

Figure 12. TreeVQA shot savings for QAOA.

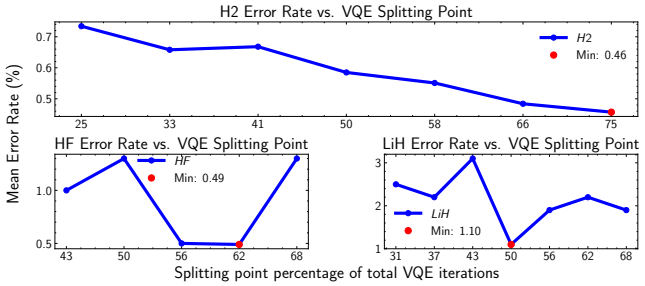

Figure 13. Analysis of splitting points and their impact on TreeVQA performance

mization level with error mitigation enabled [38, 45]. The COBYLA optimizer is used instead of SPSA, as SPSA converges too slowly under noise. As shown in Table 2, TreeVQA achieves substantial shot savings even in noisy conditions.

8.8 TreeVQA Benefits for QAOA

To demonstrate TreeVQA's applicability beyond VQE, we evaluate its performance on QAOA for MaxCut problems using the IEEE-14 bus system (see Table 7.1). This system is modeled as a 14-node undirected graph. We consider three load scale ranges: 0.5 to 1.5, 0.8 to 1.2, and 0.9 to 1.1, representing increasing levels of similarity among problem instances, from extreme planning scenarios to typical operational variations and small forecasting errors. For each range, we generate 10 graphs with edge weights determined by equally spaced load scales, and solve all 10 MaxCut instances jointly with a single TreeVQA run. For all graphs in our experiments, we use the state-of-the-art classical QAOA initialization method Red-QAOA [68]. Red-QAOA applies a graph-pooling technique to generate the initial state for unweighted MaxCut problems. In our case, since the graphs are isomorphic and differ only in edge weights, the initial state remains the same for all.

To quantify graph similarity within each range, we compute the edge weight variance (purple bars in Figure 12). All graphs are aligned to a common node ordering, and the variance is calculated as the average squared deviation of each graph's edge weights from the mean graph. As shown in Figure 12, TreeVQA achieves over 20× shot savings (blue bars) when problem instances are more similar (lower variance), highlighting its ability to exploit shared structure across related tasks. Even as the variance increases, benefits remain significant (greater than 10×). These results confirm that TreeVQA's benefits extend to combinatorial optimization and that leveraging task similarity can significantly reduce quantum resource requirements.

9 Discussion

9.1 Hyperparameter Analysis

We examine key TreeVQA hyperparameters—window size and threshold—both of which will eventually affect the splitting timing. Thus we will first analyze the splitting timing effect. To further help the analysis, we introduce the *Tree Critical Depth*, defined as the longest path from the root to a leaf in the TreeVQA tree (see Figure 2b). This metric serves as a proxy for splitting time, with more effective splitting typically resulting in shallower trees.

Analysis of Cluster Splitting Timing: Cluster splitting is central to TreeVQA's performance, so we study how its timing affects VQE accuracy and efficiency. We conduct experiments on three chemistry benchmarks and manually determine the splitting time, with each point in Figure 13 showing the final error rate when splitting occurs at different stages of optimization (expressed as a percentage of total iterations). We found that the optimal timing is application-dependent (motivating an intelligent controller) but typically occurs mid-optimization (the H_2 benchmark performs best with later splitting due to its smaller problem size, which reduces the risk of overfitting even in later stages). Splitting too early prevents subproblems from exploiting shared parameter regions, wasting resources, while splitting too late causes overfitting to the mixed Hamiltonian, requiring extra shots to escape local minima. Both cases yield fidelity losses of several percentage points under a fixed shot budget, large enough to compromise chemical accuracy.

Window Size Analysis: The window size determines how many iterations are averaged to estimate the gradient slope for convergence detection (Sec. 5.2.2). Figure 14 shows how this hyperparameter affects TreeVQA's performance. Small windows make the algorithm noise-sensitive, causing premature splitting, while overly large windows delay needed splits and risk overfitting. Sweep experiments show that a moderate window of 0.01%–0.02% of total iterations strikes the best balance, capturing convergence patterns while remaining responsive. This empirically yields up to a 5× reduction in VQA error compared to other choices. While adaptive windowing may offer further gains, we leave it to future work.

Splitting Threshold Analysis: The threshold parameter controls the aggressiveness of splitting, with larger values making splits less frequent. Sweeping a logarithmic range of thresholds reveals an optimal middle ground, achieving up to a 5× reduction in overall VQA error under a fixed shot

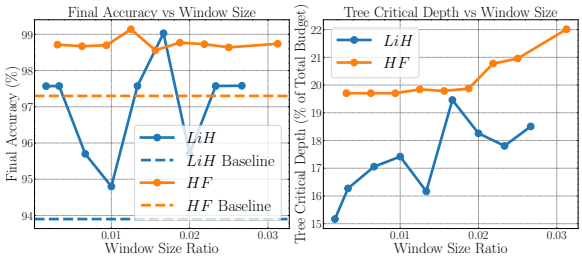

Figure 14. Window size analysis

budget. Excessively high thresholds delay splitting, causing overfitting to the mixed Hamiltonian, while low thresholds trigger premature splits and waste shots. In practice, suitable thresholds can be selected by profiling gradient slopes, as optimal values depend on Hamiltonian structure and optimizer dynamics. Overall, even with moderate hyperparameter choices, TreeVQA consistently provides significant shot savings and high accuracy, demonstrating robustness compared to baseline VQE.

9.2 Expanding TreeVQA scope

Beyond the Target Applications: TreeVQA's hierarchical clustering and adaptive shot allocation naturally extend to Quantum Machine Learning (QML). Penalty terms in QML loss functions can play roles analogous to bond lengths or magnetic fields, enabling efficient training of multiple models on the same dataset with reduced quantum costs.

Beyond SPSA and COBYLA: TreeVQA is compatible with any optimizer, requiring only cost function evaluations. It can also leverage optimizer-specific behavior — for instance, using Implicit Filtering's adaptive step sizes to adjust cluster granularity dynamically: coarser grouping during broad exploration and finer clustering when high precision is needed.

Beyond the NISQ Era: Although VQAs are often associated with NISQ due to their noise resilience, they remain compelling because of their practical applications and strong theoretical support [71]. VQAs are also relevant in the early fault-tolerant (EFT) regime, where limited quantum error correction still yields logical error rates too high for algorithms like QPE or Shor's algorithm. Consequently, recent efforts have extended NISQ-like methods into the EFT regime, including work on uncorrected RZ gates with error-corrected Cliffords [31] and VQAs on partially error-corrected devices [10, 35]. Since TreeVQA operates at a higher level of abstraction, it can be applied to EFT systems regardless of the underlying gate set (e.g., Clifford+T or Clifford+RZ), enabling significant resource savings in this emerging regime.

10 Related Work

One class of techniques reduces the number of Pauli measurements by grouping Pauli strings that can be measured simultaneously—either mutually qubit-wise commuting terms [20] or non-commuting terms that are transformed into compatible unitary sub-Hamiltonians [70]. ADAPT-VQE [24] also truncates Hamiltonian terms based on their contribution to the energy, though it was originally designed for circuit-depth reduction. These approaches are orthogonal to ours and can be integrated with TreeVQA; however, they introduce their own trade-offs, such as complex grouping heuristics or increased circuit depth due to additional unitary transformations.

Another line of work focuses on reducing the number of optimization iterations per individual task, for example, by warm-starting VQAs with classical surrogates [52] or by reducing problems to simpler forms [68]. Our evaluation in subsection 8.5 and subsection 8.8 demonstrates that TreeVQA can seamlessly combine with these methods to achieve additional benefits. However, these works do not address the multi-task structure that is inherent in real-world applications, where multiple related problems must be solved.

To our knowledge, the only prior work that explicitly targets multi-task VQE is Meta-VQE [7]. Meta-VQE designs a specialized ansatz that embeds Hamiltonian parameters directly into the circuit structure to obtain improved parameter initializations across related tasks. However, it is limited to that specific ansatz family and lacks dynamic monitoring or adaptive splitting mechanisms, meaning it cannot exploit cross-task similarity at the fine-grained level required to achieve the substantial shot savings demonstrated by TreeVQA.

11 Conclusion

We introduced TreeVQA, a hierarchical framework that significantly reduces quantum resource consumption for variational quantum algorithms (VQAs). By dynamically clustering similar Hamiltonians and adaptively allocating shots, TreeVQA achieves orders-of-magnitude measurement savings without compromising accuracy. This approach addresses the core challenge of costly quantum resources, bringing VQAs closer to practical applications. As quantum hardware advances, TreeVQA's scalable design remains relevant, supporting efficient execution in future HPC+Quantum systems and fault-tolerant architectures.

12 Acknowledgment

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Accelerated Research in Quantum Computing under Award Number DE-SC0025633. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award ASCR-ERCAP0033197.

References

[1] [n. d.]. EfficientSU2 ansatz (Qiskit Circuit Library). https://quantum.cloud.ibm.com/docs/api/qiskit/qiskit.circuit.library.EfficientSU2. Accessed: 2025-08-19.

[2] Boaz Barak, Ankur Moitra, Ryan O'Donnell, Prasad Raghavendra, Oded Regev, David Steurer, Luca Trevisan, Aravindan Vijayaraghavan, David Witmer, and John Wright. 2015. Beating the random assignment on constraint satisfaction problems of bounded degree. *arXiv* (Aug. 2015). https://doi.org/10.48550/arXiv.1505.03424

[3] Ewout van den Berg, Zlatko K. Minev, Abhinav Kandala, and Kristan Temme. 2022. Probabilistic error cancellation with sparse Pauli-Lindblad models on noisy quantum processors. https://doi.org/10.48550/ARXIV.2201.09866

[4] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. 2017. Quantum machine learning. *Nature* 549, 7671 (2017), 195–202.

[5] M. Born and V. Fock. 1928. Beweis des Adiabatensatzes. *Zeitschrift für Physik* 51, 3-4 (1928), 165–180. https://doi.org/10.1007/BF01343193

[6] Matthias C. Caro. 2024. Learning Quantum Processes and Hamiltonians via the Pauli Transfer Matrix. *ACM Transactions on Quantum Computing* 5, 2, Article 14 (June 2024), 53 pages. https://doi.org/10.1145/3670418

[7] Alba Cervera-Lierta, Jakob S. Kottmann, and Alán Aspuru-Guzik. 2021. Meta-Variational Quantum Eigensolver: Learning Energy Profiles of Parameterized Hamiltonians for Quantum Simulation. *PRX Quantum* 2, 2 (May 2021). https://doi.org/10.1103/prxquantum.2.020329

[8] Chi-Ning Chou, Peter J. Love, Juspreet Singh Sandhu, and Jonathan Shi. 2022. Limitations of Local Quantum Algorithms on Random Max-k-XOR and Beyond. https://doi.org/10.48550/arXiv.2108.06049

[9] Siddharth Dangwal, Gokul Subramanian Ravi, Poulami Das, Kaitlin N. Smith, Jonathan M. Baker, and Frederic T. Chong. 2023. VarSaw: Application-tailored Measurement Error Mitigation for Variational Quantum Algorithms. arXiv:2306.06027 [quant-ph]

[10] Siddharth Dangwal, Suhas Vittal, Lennart Maximillian Seifert, Frederic T. Chong, and Gokul Subramanian Ravi. 2025. Variational Quantum Algorithms in the era of Early Fault Tolerance. arXiv:2503.20963 [quant-ph] https://arxiv.org/abs/2503.20963

[11] Poulami Das, Swamit Tannu, and Moinuddin Qureshi. 2021. JigSaw: Boosting Fidelity of NISQ Programs via Measurement Subsetting. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture* (Virtual Event, Greece) *(MICRO '21)*. Association for Computing Machinery, New York, NY, USA, 937–949. https://doi.org/10.1145/3466752.3480044

[12] The Qiskit Nature developers and contributors. 2023. Qiskit Nature 0.6.0. https://doi.org/10.5281/zenodo.7828768

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

[14] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2015. A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem. *arXiv* (June 2015). https://doi.org/10.48550/arXiv.1412.6062

[15] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Leo Zhou. 2022. The Quantum Approximate Optimization Algorithm and the Sherrington-Kirkpatrick Model at Infinite Size. *Quantum* 6 (July 2022), 759. https://doi.org/10.22331/q-2022-07-07-759

[16] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. 2012. Surface codes: Towards practical large-scale quantum computation. *Physical Review A* 86, 3 (sep 2012). https://doi.org/10.1103/physreva.86.032324

[17] Eduardo Fradkin. 2013. *Field Theories of Condensed Matter Physics* (2 ed.). Cambridge University Press.

[18] Tudor Giurgica-Tiron, Yousef Hindy, Ryan LaRose, Andrea Mari, and William J Zeng. 2020. Digital zero noise extrapolation for quantum error mitigation. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 306–316.

[19] Fred Glover, Gary Kochenberger, and Yu Du. 2019. A Tutorial on Formulating and Using QUBO Models. *arXiv* (Nov. 2019). https://doi.org/10.48550/arXiv.1811.11538

[20] Pranav Gokhale, Olivia Angiuli, Yongshan Ding, Kaiwen Gui, Teague Tomesh, Martin Suchara, Margaret Martonosi, and Frederic T. Chong. 2020. $O(N^3)$ Measurement Cost for Variational Quantum Eigensolver on Molecular Hamiltonians. *IEEE Transactions on Quantum Engineering* 1 (2020), 1–24. https://doi.org/10.1109/TQE.2020.3035814

[21] Pranav Gokhale, Yongshan Ding, Thomas Propson, Christopher Winkler, Nelson Leung, Yunong Shi, David I. Schuster, Henry Hoffmann, and Frederic T. Chong. 2019. Partial Compilation of Variational Algorithms for Noisy Intermediate-Scale Quantum Machines. *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (Oct 2019). https://doi.org/10.1145/3352460.3358313

[22] Google. [n. d.]. Google Quantum Computing Roadmap. https://quantumai.google/roadmap. Accessed: 2024-11-22.

[23] Harper R. Grimsley, Sophia E. Economou, Edwin Barnes, and Nicholas J. Mayhall. 2019. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature Communications* 10, 1 (July 2019). https://doi.org/10.1038/s41467-019-10988-2

[24] Harper R. Grimsley, Sophia E. Economou, Edwin Barnes, and Nicholas J. Mayhall. 2019. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature Communications* 10, 1 (Jul 2019). https://doi.org/10.1038/s41467-019-10988-2

[25] Lov K Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 212–219.

[26] Andi Gu, Angus Lowe, Pavel A Dub, Patrick J Coles, and Andrew Arrasmith. 2021. Adaptive shot allocation for fast convergence in variational quantum algorithms. *arXiv preprint arXiv:2108.10434* (2021).

[27] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G. Rieffel, Davide Venturelli, and Rupak Biswas. 2019. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms* 12, 2 (Feb. 2019), 34. https://doi.org/10.3390/a12020034

[28] M. B. Hastings. 2019. Classical and Quantum Bounded Depth Approximation Algorithms. *arXiv* (Aug. 2019). https://doi.org/10.48550/arXiv.1905.07047

[29] Rebekah Herrman, Phillip C. Lotshaw, James Ostrowski, Travis S. Humble, and George Siopsis. 2021. Multi-angle Quantum Approximate Optimization Algorithm. arXiv:2109.11455 [quant-ph] https://arxiv.org/abs/2109.11455

[30] IBM. [n. d.]. IBM Quantum Computing Roadmap. https://www.ibm.com/roadmaps/quantum. Accessed: 2024-11-22.

[31] Refaat Ismail, I-Chi Chen, Chen Zhao, Ronen Weiss, Fangli Liu, Hengyun Zhou, Sheng-Tao Wang, Andrew Sornborger, and Milan Kornjača. 2025. Transversal STAR architecture for megaquop-scale quantum simulation with neutral atoms. arXiv:2509.18294 [quant-ph] https://arxiv.org/abs/2509.18294

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

[33] Hang Jing, Ye Wang, and Yan Li. 2023. Data-driven quantum approximate optimization algorithm for power systems. *Communications Engineering* 2, 1 (2023), Article 12. https://doi.org/10.1038/s44172-023-00061-8

[34] Abhinav Kandala, Kristan Temme, Antonio D. Córcoles, Antonio Mezzacapo, Jerry M. Chow, and Jay M. Gambetta. 2019. Error mitigation extends the computational reach of a noisy quantum processor. *Nature* 567, 7749 (2019), 491–495. https://doi.org/10.1038/s41586-019-1040-7

[35] Amara Katabarwa, Katerina Gratsea, Athena Caesura, and Peter D.

Johnson. 2024. Early Fault-Tolerant Quantum Computing. *PRX Quantum* 5 (Jun 2024), 020101. Issue 2. https://doi.org/10.1103/PRXQuantum.5.020101

[36] Youngseok Kim, Andrew Eddins, Sajant Anand, Ken Xuan Wei, Ewout van den Berg, Sami Rosenblatt, Hasan Nayfeh, Yantao Wu, Michael Zaletel, Kristan Temme, and Abhinav Kandala. 2023. Evidence for the Utility of Quantum Computing before Fault Tolerance. *Nature* 618, 7965 (June 2023), 500–505. https://doi.org/10.1038/s41586-023-06096-3

[37] Wim Lavrijsen, Ana Tudor, Juliane Müller, Costin Iancu, and Wibe de Jong. 2020. Classical Optimizers for Noisy Intermediate-Scale Quantum Devices. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. 267–277. https://doi.org/10.1109/QCE49297.2020.00041

[38] Gushu Li, Yufei Ding, and Yuan Xie. 2019. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*. 1001–1014. https://doi.org/10.1145/3297858.3304023

[39] LibreTexts. 2025. Physical Chemistry LibreTexts. https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry. Accessed: 2025-08-16.

[40] Cedric Yen-Yu Lin and Yechao Zhu. 2016. Performance of QAOA on Typical Instances of Constraint Satisfaction Problems with Bounded Degree. *arXiv* (Jan. 2016). https://doi.org/10.48550/arXiv.1601.01744

[41] Kunal Marwaha. 2021. Local classical MAX-CUT algorithm outperforms $p=2$ QAOA on high-girth regular graphs. *Quantum* 5 (April 2021). https://doi.org/10.22331/q-2021-04-20-437

[42] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. 2016. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics* 18, 2 (2016), 023023.

[43] Layne Morsch. 2023. *Intermediate Physical Organic Chemistry*. LibreTexts. https://chem.libretexts.org/Bookshelves/Organic_Chemistry/Intermediate_Physical_Organic_(Morsch) Accessed: 2025-04-10.

[44] Mario Motta, Kevin J. Sung, K. Birgitta Whaley, Martin Head-Gordon, and James Shee. 2023. Bridging physical intuition and hardware efficiency for correlated electronic states: the local unitary cluster Jastrow ansatz for electronic structure. *Chemical Science* 14 (2023), 11213–11227. https://doi.org/10.1039/D3SC02516K

[45] Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. 2019. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 1015–1029.

[46] Ken N. Okada, Keita Osaki, Kosuke Mitarai, and Keisuke Fujii. 2023. Classically optimized variational quantum eigensolver with applications to topological phases. *Phys. Rev. Res.* 5 (Dec 2023), 043217. Issue 4. https://doi.org/10.1103/PhysRevResearch.5.043217

[47] Joe O'Gorman and Earl T. Campbell. 2017. Quantum computation with realistic magic-state factories. *Physical Review A* 95, 3 (Mar 2017). https://doi.org/10.1103/physreva.95.032338

[48] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. 2014. A variational eigenvalue solver on a photonic quantum processor. *Nature communications* 5 (2014), 4213.

[49] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (2018), 79.

[50] Qiskit Nature Development Team. 2023. LinearMapper — Qiskit Nature 0.6.0 documentation. https://qiskit-community.github.io/qiskit-nature/stubs/qiskit_nature.second_q.mappers.LinearMapper.html. Accessed: 2025-04-10.

[51] Gokul Subramanian Ravi, Pranav Gokhale, Yi Ding, William M. Kirby, Kaitlin N. Smith, Jonathan M. Baker, Peter J. Love, Henry Hoffmann, Kenneth R. Brown, and Frederic T. Chong. 2022. CAFQA: Clifford Ansatz For Quantum Accuracy. https://doi.org/10.48550/ARXIV.2202.12924

[52] Gokul Subramanian Ravi, Pranav Gokhale, Yi Ding, William M. Kirby, Kaitlin N. Smith, Jonathan M. Baker, Peter J. Love, Henry Hoffmann, Kenneth R. Brown, and Frederic T. Chong. 2023. CAFQA: A classical simulation bootstrap for variational quantum algorithms. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. https://doi.org/10.48550/ARXIV.2202.12924

[53] Anton Robert, Panagiotis Kl Barkoutsos, Stefan Woerner, and Ivano Tavernelli. 2021. Resource-efficient quantum algorithm for protein folding. *npj Quantum Information* 7, 1 (2021), 38.

[54] Manuel S. Rudolph, Tyson Jones, Yanting Teng, Armando Angrisani, and Zoë Holmes. 2025. Pauli Propagation: A Computational Framework for Simulating Quantum Systems. arXiv:2505.21606 [quant-ph] https://arxiv.org/abs/2505.21606

[55] J. J. Sakurai and Jim Napolitano. 2020. *Modern Quantum Mechanics* (3 ed.). Cambridge University Press.

[56] Ruslan Shaydulin, Phillip C. Lotshaw, Jeffrey Larson, James Ostrowski, and Travis S. Humble. 2023. Parameter Transfer for Quantum Approximate Optimization of Weighted MaxCut. *ACM Transactions on Quantum Computing* 4 (Sept. 2023). https://doi.org/10.1145/3584706

[57] Ruslan Shaydulin, Kunal Marwaha, Jonathan Wurtz, and Phillip C. Lotshaw. 2021. QAOAKit: A Toolkit for Reproducible Study, Application, and Verification of the QAOA. *arXiv:2110.05555 [quant-ph]* (Nov. 2021).

[58] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26, 5 (Oct 1997), 1484–1509. https://doi.org/10.1137/s0097539795293172

[59] James Spall. 2001. An Overview of the Simultaneous Perturbation Method for Efficient Optimization. (02 2001).

[60] Michael Streif and Martin Leib. 2020. Training the quantum approximate optimization algorithm without access to a quantum processing unit. *IOP* 5, 3 (May 2020), 034008. https://doi.org/10.1088/2058-9565/ab8c2b Publisher: IOP Publishing.

[61] Wen-Yu Su, Feng Hu, Chen Cheng, and Nvsen Ma. 2023. Berezinskii-Kosterlitz-Thouless phase transitions in a kagome spin ice by a quantifying Monte Carlo process: Distribution of Hamming distances. *Phys. Rev. B* 108 (Oct 2023), 134422. Issue 13. https://doi.org/10.1103/PhysRevB.108.134422

[62] Qiming Sun, Timothy C. Berkelbach, Nick S. Blunt, George H. Booth, Sheng Guo, Zhendong Li, Junzi Liu, James D. McClain, Elvira R. Sayfutyarova, Sandeep Sharma, Sebastian Wouters, and Garnet Kin-Lic Chan. 2018. PySCF: the Python-based simulations of chemistry framework. *WIREs Computational Molecular Science* 8, 1 (2018), e1340. https://doi.org/10.1002/wcms.1340 arXiv:https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wcms.1340

[63] Attila Szabo and Neil S Ostlund. 1996. *Modern quantum chemistry: introduction to advanced electronic structure theory*. Courier Corporation.

[64] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, and Jonathan Tennyson. 2022. The Variational Quantum Eigensolver: A review of methods and best practices. *Physics Reports* 986 (2022), 1–128. https://doi.org/10.1016/j.physrep.2022.08.003 The Variational Quantum Eigensolver: a review of methods and best practices.

[65] Andrew Tranter, Sarah Sofia, Jake Seeley, Michael Kaicher, Jarrod McClean, Ryan Babbush, Peter V. Coveney, Florian Mintert, Frank Wilhelm, and Peter J. Love. 2015. The Bravyi–Kitaev transformation: Properties and applications. *International Journal of Quantum Chemistry* 115, 19 (2015), 1431–1441. https://doi.org/10.1002/qua.24969 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/qua.24969

[66] M. Venkateswara Reddy, Bishnu Prasad Muni, and A.V.R.S. Sarma. 2016.

Enhancement of voltage profile for IEEE 14 bus system with inter line power flow controller. In *2016 Biennial International Conference on Power and Energy Systems: Towards Sustainable Energy (PESTSE)*. 1–5. https://doi.org/10.1109/PESTSE.2016.7516503

[67] Ulrike von Luxburg. 2007. A Tutorial on Spectral Clustering. arXiv:0711.0189 [cs.DS] https://arxiv.org/abs/0711.0189

[68] Meng Wang, Bo Fang, Ang Li, and Prashant J. Nair. 2024. Red-QAOA: Efficient Variational Optimization through Circuit Reduction. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (La Jolla, CA, USA) *(ASPLOS '24)*. Association for Computing Machinery, New York, NY, USA, 980–998. https://doi.org/10.1145/3620665.3640363

[69] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. 2018. Quantum Approximate Optimization Algorithm for MaxCut: A Fermionic View. *Physical Review A* 97, 2 (Feb. 2018), 022304. https://doi.org/10.1103/PhysRevA.97.022304

[70] Tzu-Ching Yen, Vladyslav Verteletskyi, and Artur F. Izmaylov. 2020. Measuring All Compatible Operators in One Series of Single-Qubit Measurements Using Unitary Transformations. *Journal of Chemical Theory and Computation* 16, 4 (2020), 2400–2409. https://doi.org/10.1021/acs.jctc.0c00008 PMID: 32150412.

[71] Zoltán Zimborás, Bálint Koczor, Zoë Holmes, Elsi-Mari Borrelli, András Gilyén, Hsin-Yuan Huang, Zhenyu Cai, Antonio Acín, Leandro Aolita, Leonardo Banchi, Fernando G. S. L. Brandão, Daniel Cavalcanti, Toby Cubitt, Sergey N. Filippov, Guillermo García-Pérez, John Goold, Orsolya Kálmán, Elica Kyoseva, Matteo A. C. Rossi, Boris Sokolov, Ivano Tavernelli, and Sabrina Maniscalco. 2025. Myths around quantum computation before full fault tolerance: What no-go theorems rule out and what they don't. arXiv:2501.05694 [quant-ph] https://arxiv.org/abs/2501.05694