PAPER • OPEN ACCESS

YAQQ: yet another quantum quantizer design space exploration of quantum gate sets using novelty search

To cite this article: Aritra Sarkar *et al* 2026 *New J. Phys.* **28** 044504

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

PAPER

YAQQ: yet another quantum quantizer design space exploration of quantum gate sets using novelty search

Aritra Sarkar[1,2,3], **Akash Kundu**[1,4,5,6], **Matthew Steinberg**[2,3], **Sibasish Mishra**[2], **Sebastiaan Fauquenot**[2], **Tamal Acharya**[1], **Jarosław A Miszczak**[5] and **Sebastian Feld**[2,3,*]

[1] Quantum Intelligence Alliance, Kolkata, India
[2] Quantum Machine Learning research group, Quantum Computing division, QuTech, Delft, The Netherlands
[3] Department of Quantum & Computer Engineering, Delft University of Technology, Delft, The Netherlands
[4] Joint Doctoral School, Silesian University of Technology, Gliwice, Poland
[5] Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Gliwice, Poland
[6] QTF Centre of Excellence, Department of Physics, University of Helsinki, Helsinki, Finland
[*] Author to whom any correspondence should be addressed.

E-mail: s.feld@tudelft.nl

Keywords: quantum compiler, gate set tomography, quantum circuits, quantum error correction

OPEN ACCESS

RECEIVED
14 January 2026

REVISED
31 March 2026

ACCEPTED FOR PUBLICATION
1 April 2026

PUBLISHED
9 April 2026

Abstract

The standard model of quantum computation is based on quantum circuits, where the number and quality of the quantum gates composing the circuit influence the runtime and fidelity of the computation. The fidelity of the decomposition of quantum algorithms, represented as unitary matrices, to bounded depth quantum circuits depends strongly on the set of gates available for the decomposition routine. To investigate this dependence, we explore the design space of discrete quantum gate sets and present a software tool for comparative analysis of quantum processing units and control protocols based on their native gates. The evaluation is conditioned on a set of unitary transformations representing target use cases on the quantum processors. The cost function considers three key factors: (i) the statistical distribution of the decomposed circuits' depth, (ii) the statistical distribution of process fidelities for the approximate decomposition, and (iii) the relative novelty of a gate set compared to other gate sets in terms of the aforementioned properties. The developed software, called yet another quantum quantizer (YAQQ), enables the discovery of an optimized set of quantum gates through this tunable joint cost function. To identify these gate sets, we use the novelty search algorithm, circuit decomposition techniques (like Solovay–Kitaev, Cartan, and quantum Shannon decomposition), and stochastic optimization to implement YAQQ within the Qiskit quantum simulator environment. YAQQ exploits reachability tradeoffs conceptually derived from quantum algorithmic information theory. Our results demonstrate the pragmatic application of identifying gate sets that are advantageous to popularly used quantum gate sets in representing quantum algorithms. Consequently, we demonstrate pragmatic use cases for YAQQ, including comparing transversal logical gate sets in quantum error correction codes and designing optimal quantum instruction sets for a benchmark suite of quantum algorithms.

1. Introduction

A crucial step in executing a quantum algorithm on a quantum processor unit (QPU) is optimizing the quantum circuit, which is represented as a sequence of native quantum logic gates. This optimization has two purposes. Firstly, in the noisy intermediate-scale quantum (NISQ) era [1, 2], the useful circuit depth is limited by decoherence and gate errors, so longer circuits reduce the fidelity of the output. Secondly, the resource advantage of quantum computation (QC) over classical models is usually expressed as asymptotic time-complexity gains [3–5], and overly long or poorly compiled circuits can delay the practical quantum-over-classical advantage to much larger problem sizes in both NISQ and fault-tolerant quantum computing (FTQC) regimes.

Classical and quantum compilers tackle this challenge using circuit optimization techniques such as algebraic rewriting, ZX-calculus [6], variational compilation [7], unitary decomposition [8], and program synthesis [9], while respecting hardware constraints like qubit connectivity, native gate sets, control-frequency sharing, and fabrication imperfections. The impact of these strategies is quantified by benchmarking suites such as QVolume [10], QPack [11], QScore [12], Quantum Resource Estimator [13], and MQT Bench [14], which typically target runtime, gate counts, or fidelity.

In this work, we instead examine a complementary but distinct question: how the *choice of native gate set* determines the decomposition fidelity and circuit depth (and thereby the runtime and decoherence) of compiled quantum algorithms. Previous work has studied universal and near-optimal gate sets, e.g. by characterizing universal generators [15], analyzing golden gates [16, 17], and tailoring native sets to specific QPUs [18–20]. These approaches typically assume a fixed hardware-native set and optimize circuits or hardware parameters around it. In this work, we instead treat the native gate set itself as a design variable.

We introduce the yet another quantum quantizer (YAQQ)[7] software tool as an implementation of the proposal. YAQQ quantifies how circuit depth and fidelity depend on the chosen gate set by benchmarking candidate one- and two-qubit gate sets on datasets of target unitaries (Haar-random, algorithm-derived, or application-specific). YAQQ focuses on small, discrete FTQC-compatible gate sets, searches for hardware-agnostic novel gate sets, and works entirely in simulation, in contrast to active-learning compilation methods [22, 23] that require repeated query of quantum hardware.

To this end, YAQQ provides a framework for comparing native gate sets using a multi-objective cost function that combines three components: (i) the statistical distribution of circuit depths required to approximate the dataset unitaries; (ii) the statistical distribution of process fidelities of these approximations; and (iii) an algorithmic information theory inspired *novelty* score that rewards gate sets whose depth and fidelity tradeoffs differ significantly from those of reference gate sets. By optimizing this tunable cost, YAQQ performs design space exploration (DSE) over gate sets, and uses a combination of novelty search [24], quantum gate decomposition techniques [25], random search, and SciPy-based optimization routines to identify gate sets with favorable reachability properties within a Qiskit-based environment [26].

We demonstrate that YAQQ can discover a native gate set that yields shallow, high-fidelity decompositions for one- and two-qubit unitaries, outperforming the commonly used fault-tolerant gate set (Hadamard, T, CX) in average process fidelity while maintaining competitive circuit depth on representative datasets. We further show how YAQQ can inform the design of energy-efficient quantum instruction set architectures (QISAs) and support comparative analyses of transversal logical gate sets in quantum error-correcting (QEC) codes, highlighting its relevance for both NISQ- and FTQC-era architecture design.

Contributions

This article makes the following contributions:

1. We introduce YAQQ, a software framework for DSE of discrete quantum gate sets, which treats the native gate set as an architectural choice and evaluates candidate sets against user-specified datasets of target unitaries via a tunable, multi-objective cost function that combines depth, fidelity, and novelty.
2. We develop an algorithmic pipeline that integrates Solovay–Kitaev, Shannon, and Cartan decompositions with stochastic search and SciPy-based optimization. This enables hardware-agnostic discovery and benchmarking of gate sets within the Qiskit transpilation stack.
3. We provide numerical evidence that YAQQ can identify gate sets that outperform standard fault-tolerant gate sets such as {H, T, CX} in average process fidelity at comparable or reduced depth across both random-unitary datasets and real-world algorithm benchmark suites. We illustrate its application in QISA design and comparative analysis of transversal gate sets in QEC codes.
4. We provide an easy-to-use YAQQ framework through PyPI at https://pypi.org/project/yaqq/. The source code and data generated in the paper are available publicly through GitHub for reproducibility.

[7] The YAQQ name is inspired by YACC [21], a compiler-compiler or compiler-generator; similarly, YAQQ synthesizes the decomposition transpilation pass for the generated gate set.

In what follows, we present some necessary background in section 2. The main design choices and algorithmic blocks of YAQQ for exploring quantum gate sets are introduced in section 3. In section 4, we discuss the results of fine-tuning the hyperparameters, discovering novel gate sets using a random dataset, and exemplary applications of YAQQ. Section 5 concludes the article.

2. Related work

In this section, we compare gate set discovery with related yet distinct research directions, namely variational compilation [22] and quantum architecture search (QAS) [27]. Thereafter, we provide a brief background on novelty search, which YAQQ employs to optimize gate sets. Variational quantum compilation methods parameterize circuits using a fixed gate alphabet with continuous parameters and optimize those parameters to approximate a target unitary or algorithmic objective [22]. This framework is extended in [28] to tackle many quantum system requirements by simultaneous optimization of multiple targets. Moreover, in [29] the authors propose a purely quantum computer-based compilation of quantum algorithms, paving the way towards quantum advantage in compilation. Similarly, QAS frameworks (we refer to [30] for a list of QAS approaches) treat the gate set as a predefined action space within reinforcement learning [31–34] or as a differentiable optimization space [27]. Simultaneously existing compilation and synthesis frameworks such as BQSKit [35, 36] and Superstaq [37] all explore circuit topologies or parameter values while assuming a fixed set of primitive operations. In contrast, YAQQ treats the gate set itself as a design variable. Rather than optimizing parameters within a fixed alphabet, YAQQ restricts the candidate gate set to a discrete, finite collection and performs design-space exploration (DSE) to identify gate subsets that minimize a given compilation cost function, such as circuit depth, approximation error, or hardware constraints. This formulation is similar to inductive language inference [38]. YAQQ's discrete gate set discovery enables the identification of effective primitive operations tailored to a specific cost model or target circuit distribution.

Novelty search [24, 39] is an evolutionary search paradigm that departs from traditional objective-driven optimization by rewarding behavioral novelty instead of direct improvement in a predefined fitness function. Formally, novelty search defines a behavior characterization, $b(x)$, for a candidate solution x, which maps the solution to a descriptor space that captures salient behavioral features. The novelty of a candidate is then quantified as the average distance in this behavior space to its k-nearest neighbors drawn from the current population and an archive of previously discovered behaviors:

$$\rho(x) = \frac{1}{k}\sum_{i=1}^{k} \mathrm{dist}(b(x), b(\mu_i)),$$

where μ_i denotes the ith nearest neighbor according to a chosen metric. Candidates with higher novelty scores are preferentially selected, encouraging the search to explore previously unvisited regions of the behavior space.

This approach has been shown to be particularly effective in domains with deceptive or sparse objective landscapes, such as robotics control, procedural content generation, and open-ended evolutionary (OEE) systems, where direct optimization can prematurely converge to local optima. In this work, we explore the application of novelty search to gate set discovery for quantum circuit decomposition. Unlike conventional compilation methods that optimize circuits over a fixed set of primitive gates, this formulation treats candidate gate sets as individuals whose behavioral descriptors may capture properties such as decomposition efficiency, approximation error profiles, or hardware compatibility across a distribution of target unitaries. To preserve the practicality of the OEE from novelty search, the intrinsic motivation [40, 41] is jointly incorporated via an externally defined cost function (reward) that captures quantum circuit metrics such as fidelity and depth. This is further detailed in section 3.3. To our knowledge, the use of novelty-driven evolutionary search to discover useful primitive gate sets has not been previously explored in quantum compilation. Such an approach aligns naturally with emerging directions in automated scientific discovery, where algorithmic systems explore large design spaces to identify structures and primitives that improve downstream tasks without requiring explicit objective shaping.

3. Methods

This section presents the core method behind our YAQQ. YAQQ is a tool for automated quantum gate set design. The primary target is to develop specifications for a quantum processor that is yet to be

manufactured (or the operating procedures of a tunable quantum computer). However, it can also be used to compare existing specifications or to perform an in-depth evaluation of a single specification.

We are interested in the space of building blocks that allow efficient gate arrangements for practical quantum applications and the circuit structures based on those blocks. For an effective DSE, five aspects of the problem formulation are crucial: (i) the abstraction level for quantum unitary control and synthesis, (ii) the ansatz used to parameterize candidate gate sets, (iii) the cost function used to evaluate gate sets, (iv) the dataset of target unitaries used for benchmarking, and (v) the circuit decomposition techniques used to compile these unitaries with a given gate set. These aspects are detailed in the remainder of this section.

3.1. Quantum unitary control and synthesis

At an abstract level, gate-based quantum computing implements an n-qubit computation as a unitary $U \in \mathbb{C}^{2^n \times 2^n}$ acting on an initial state (typically $|0\rangle^{\otimes n}$), followed by measurement in the computational Z-basis. One can, in principle, optimize U at three levels: pulse-level control signals, sequences of local parametric gates, and sequences of local gates from a finite discrete set. YAQQ operates at the gate level and does not perform pulse-level optimal control. We assume each gate has unit time cost. Pulse-level refinements and gate-time inhomogeneities can be incorporated afterwards, e.g. using [42], to obtain hardware-specific time or energy estimates for the discovered gate sets.

Parametric gates. A gate set is computationally universal if any QC can be efficiently expressed using its elements. Any U can be decomposed into k-local gates that act on at most k qubits, independently of physical connectivity. Routing enforces physical locality and adds a worst-case constant-factor depth overhead that depends on the device size [43]. The minimal k for universal QC is of both theoretical and practical interest. For example, the 3-qubit Deutsch gate $D(\theta)$ is universal, while $k=2$ suffices with a CX gate plus arbitrary single-qubit rotations about two orthogonal axes [44]. However, an exact decomposition of a generic n-qubit unitary requires at least $\frac{1}{4}(4^n - 3n - 1)$ CX gates [45, 46], and similar exponential worst-case bounds hold for classical 2-local Boolean circuits [47]. Quantum Shannon decomposition (QSD) [48] provides a standard recursive decomposition of U into 2-local gates, and we use QSD within YAQQ as a first step for multi-qubit DSE (see also section 3.2).

Finite set of discrete gates. After routing, current QPUs typically realize universal QC using a fixed 2-qubit entangling gate (e.g. CX or CZ) with arbitrary 1-qubit rotations around the X, Y, and Z axes. In YAQQ, we nevertheless restrict ourselves to finite discrete gate sets, for three main reasons.

Firstly, full quantum characterization is costly. Techniques such as gate set tomography [49] are typically applied only to a small set of discrete rotation angles (e.g. 90° and 45°), and compilation pipelines in practice trust only this characterized subset.

Secondly, FTQC uses QEC codes, where a logical qubit is encoded into many physical qubits. Logical gates are implemented as local operations on the code, and fault-tolerant logical operations are often required to be transversal. Transversality is usually proven for specific discrete gates and specific codes. Due to the Eastin–Knill theorem [50], no set of transversal gates can be universal; additional resource states (e.g. magic states) are needed. Consequently, an FTQC stack is built around a small discrete logical gate set and resource states, and any U must be decomposed into this discrete set as efficiently as possible.

Thirdly, even for NISQ hardware without QEC, control is effectively discrete. Variational quantum algorithms rely on continuous parameters [51], but the effective precision of gate angles is limited by data types in the programming language, instruction bandwidth in the microarchitecture, and DAC resolution in the control electronics. As the system size grows, the reachable volume of the Hilbert space within bounded depth and error shrinks rapidly, even when the underlying model uses parametric rotations.

The space of universal gate sets in quantum computing is uncountably infinite, a stronger analog of the ubiquity of universality in classical computation [52]. The Solovay–Kitaev theorem (SKT) shows that for a finite generating set $\mathcal{G} \subset SU(2)$ whose group closure is dense, any single-qubit unitary U can be approximated to precision $\epsilon > 0$ by a sequence S over $\mathcal{G}$ of length $O(\log^{\alpha}(1/\epsilon))$ [25], for some constant α, and that such a sequence can be found in time $O(\log^{\beta}(1/\epsilon))$ for some β [25, 53]. The generators of the associated Lie algebra $\mathfrak{su}(d)$ are Hermitian Hamiltonians (e.g. Pauli matrices for $\mathfrak{su}(2)$ or Gell–Mann matrices for $\mathfrak{su}(3)$).

For multi-qubit unitaries, both the length of the decomposed circuit and the classical runtime of SK-based algorithms grow exponentially with the number of qubits [25]. Moreover, SK-based sequences

typically explore only a sparse subset of all possible approximations and are often far from length-optimal [54]. For this reason, YAQQ combines several decomposition techniques: (i) QSD to reduce n-qubit unitaries to 2-local gates, (ii) Cartan (KAK) decomposition to translate between CX and other 2-qubit entangling gates in the candidate gate set, (iii) Solovay–Kitaev decomposition (SKD) for single-qubit decompositions, and (iv) random decomposition (RD) as a stochastic alternative. RD samples random sequences over the candidate gate set and keeps the one with highest fidelity; it extends easily to higher-dimensional Hilbert spaces, at the cost of increasing the number of trials. YAQQ exposes these options so that gate-set performance can be evaluated under different decomposition strategies; quantitative comparisons are reported in section 4.

3.2. Novel gate set ansatz

The gate set definition is an important step in pruning the search space. It is well known that 2-local gates are universal for QC [55]. Circuits containing higher-order gates can be decomposed into 1- and 2-qubit gates, e.g. via QSD. From an engineering perspective, almost all quantum processors also expose modular native 1- and 2-qubit gates, with 3-qubit native gates still experimental and typically low-fidelity. In this work, we therefore search over gate sets composed of 1- and 2-qubit gates. Depending on the gate type, the search can be parametric or random. Additionally, some standard constant gates can be included, but are not optimized. YAQQ provides several built-in gates, listed in table 4 in appendix B.

A general n-qubit gate has $2^{2n}-1$ real parameters; thus, a single-qubit gate has 3 free parameters, and a two-qubit gate has 15. A general single-qubit gate can be specified using the P1 gate (equivalent to IBM's U3 gate) as

$$\mathrm{P1}_{\vec{a}} = \mathrm{P1}_{a_1,a_2,a_3} = \begin{bmatrix} \cos(a_1/2) & -e^{ia_3}\sin(a_1/2) \\ e^{ia_2}\sin(a_1/2) & e^{i(a_2+a_3)}\cos(a_1/2) \end{bmatrix}. \tag{1}$$

For a 2-qubit gate, the 15 parameters can be realized as a non-local unitary NL2 on the Weyl chamber with three coordinates, sandwiched by four local P1 gates (one before and one after on each qubit), giving $3+4\times 3$ parameters in total. The NL2 gate (often referred to as the 2-qubit canonical gate) is defined as

$$\mathrm{NL2}_{\vec{t}} = \mathrm{NL2}_{t_x,t_y,t_z} = \exp\left(-i\frac{\pi}{2}\left(t_x X\otimes X + t_y Y\otimes Y + t_z Z\otimes Z\right)\right), \tag{2}$$

where $X\otimes X$, $Y\otimes Y$, and $Z\otimes Z$ have their standard 4×4 matrix representations.

All other 1- and 2-qubit gates can be expressed in terms of these two generic gates; see [56] for explicit constructions. The user specifies a gate-set ansatz in terms of these building blocks, e.g. $\{\mathrm{R1},\mathrm{R1},\mathrm{CX2}\}$ or $\{\mathrm{P1},\mathrm{T1},\mathrm{SPE2}\}$. Note that we do not explicitly enforce universality; YAQQ can therefore also be used for QEC applications, where many operations lie in the non-universal Clifford group. Once a gate-set ansatz is chosen, the total number of free parameters is fixed. YAQQ then applies the selected stochastic search or optimization routine to these parameters to generate novel complementary gate sets. Thus, the search space of YAQQ for gate discovery is continuous and parametric, while the optimized gate set is finite and discrete.

3.3. Hardware-agnostic novelty search with cost function

YAQQ searches over candidate gate sets, as defined in section 3.2, using a tunable joint cost function defined on a dataset of target unitaries (section 3.4). For two gate sets gs_1 and gs_2 and a dataset ds, the cost is

$$c(gs_2 \mid gs_1, ds) = \underbrace{w_{\mathrm{apf}}c_{\mathrm{apf}} + w_{\mathrm{npf}}c_{\mathrm{npf}} + w_{\mathrm{acd}}c_{\mathrm{acd}} + w_{\mathrm{ncd}}c_{\mathrm{ncd}}}_{\text{quantum information}} + \underbrace{w_{\mathrm{agf}}c_{\mathrm{agf}}}_{\text{quantum engineering}}. \tag{3}$$

The first four terms measure how gs_2 compares to gs_1 in terms of approximation fidelity and circuit depth. The metric c_{apf} is the improvement in average process fidelity when decomposing the dataset unitaries with gs_2 instead of gs_1. Process fidelity between two circuits qc_1 and qc_2 is computed in superoperator form as $\mathrm{PF}(qc_1,qc_2) = \mathrm{Tr}\left[\chi^{\dagger}_{qc_2}\chi_{qc_1}\right]/d^2$, where χ is a superoperator (e.g. Choi) representation of the corresponding unitary and d is the Hilbert space dimension. The metric c_{acd} is the improvement in average circuit depth, where depth counts each gate equally (no ancilla are added) and thus provides an upper bound on runtime.

The metrics c_{npf} and c_{ncd} implement *novelty search* (NS) on fidelity and depth profiles. In YAQQ, we treat the pattern of process fidelities (or depths) across the dataset as a 'behavioral signature' of a gate

set. For each data point, we compare whether gate set gs_2 performs better or worse than gs_1 (e.g. by centering fidelities around 0.5), and then measure the distance between these signatures and normalize it by the average fidelity or depth. The scores c_{npf} and c_{ncd} are higher when gs_2 achieves a *different* fidelity or depth trend than gs_1, while still allowing the weights w_{apf} and w_{acd} to encode preference for overall performance. Together, these four terms allow one to tune the trade-off between exploitation (better average fidelity/depth) and exploration (novel behavior) on the dataset [41]. To our knowledge, this is the first explicit use of NS in quantum compilation or automata-style models, although related intrinsic-motivation ideas have appeared in quantum-circuit discovery and synthesis [57–59].

The final term c_{agf} incorporates engineering constraints. While the first four metrics are information-theoretic, the practical value of a gate set also depends on how easy its gates are to fabricate, calibrate, and control on a given platform. The metric c_{agf} is defined as a bias on average gate fabrication or control difficulty and can be instantiated per target QPU. By adjusting the weights $w_{\text{apf}}, w_{\text{npf}}, w_{\text{acd}}, w_{\text{ncd}}, w_{\text{agf}}$, YAQQ can be steered from purely information-driven search (depth and fidelity only) toward hardware-aware search that also accounts for platform-specific implementation costs.

We want to highlight that the five weights $\{w_{\text{apf}}, w_{\text{npf}}, w_{\text{acd}}, w_{\text{ncd}}, w_{\text{agf}}\}$ are the primary control the user has to tune YAQQ according to the specific project requirement. Rather than prescribing a single universal configuration, YAQQ exposes these weights as hyperparameters: a practitioner focused on fidelity can up-weight w_{apf}, one targeting shallow circuits can emphasize w_{acd}, and one requiring gate-set diversity can increase w_{npf} or w_{ncd}. The systematic hyperparameter study in section 4.1.1 identifies the optimal setting for fidelity-driven gate-set discovery and demonstrates empirically that the cost function is robust to moderate weight perturbations around this optimum. We deliberately do not advocate a single fixed choice because the optimal balance is inherently application-dependent and the weight space itself constitutes a meaningful design dimension of YAQQ.

Conceptually, using a finite gate set for quantum compilation differs from the classical Boolean case. Classical universal sets can exactly represent all Boolean functions, whereas a finite universal quantum set only approximates an uncountable family of unitaries. This induces a trade-off between approximation quality and description length that is captured in quantum algorithmic information theory, in particular quantum Kolmogorov complexity [60–62]. Counting arguments show that, for any fixed encoding scheme with b classical bits, most states and unitaries are incompressible [62, 63]. In the YAQQ context, this suggests that different gate sets will encode different regions of Hilbert space more or less efficiently. By explicitly rewarding novelty in depth–fidelity behavior over a dataset, the cost function biases the search toward gate sets that provide complementary encodings of the same target unitaries. The precise implementation of these metrics is detailed in section 4.

3.4. Dataset of unitaries for benchmarking

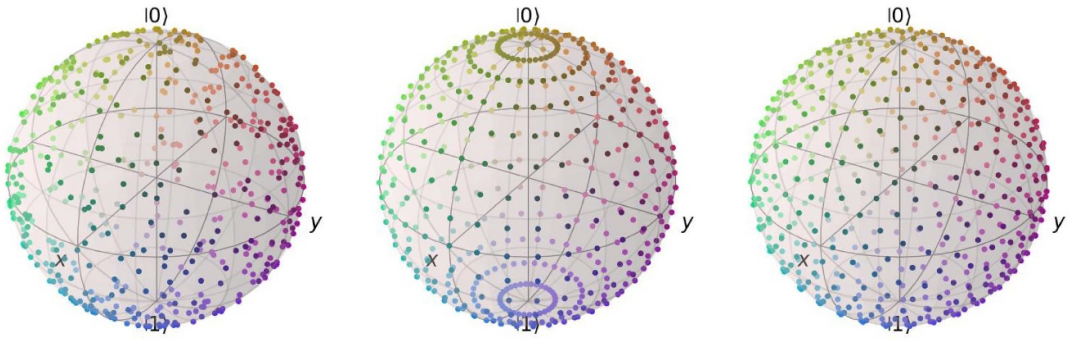
In contrast to circuit compilation, YAQQ is not concerned with optimizing one specific unitary. We require a gate set to work reasonably well for most algorithms in terms of both expressibility and fidelity. Thus, the comparative evaluation of gate sets is always conditioned on a dataset of unitaries. This dataset can be a general Haar-random set, or a curated set tailored to specific algorithms or state-preparation tasks. The dataset strongly influences YAQQ's scores for two reasons: interpretability and reachability.

From an interpretability perspective, typical gate sets in quantum computing are based on physically meaningful operations, such as bit-flip, phase-flip, and controlled-NOT. These gates serve as composable blocks for reasoning about and designing algorithms. While YAQQ focuses on QPU-native gate sets rather than 'nice' algorithmic gates, it can still be used to *discover* useful building blocks. Consequently, the gate-set YAQQ search overlaps with quantum circuit element discovery [31, 57, 58] and quantum program synthesis [64, 65]. In section 4.2.2, we discuss how to use YAQQ for interpretable gate-set discovery and for designing efficient QISA.

From a reachability perspective, all universal gate sets are asymptotically equivalent in expressivity with unbounded resources [44], but they can differ dramatically under bounded depth and error. We distinguish:

- *Expressibility*: the ability to represent arbitrary unitaries with an unbounded number of gates.
- *Reachability*: the subset of unitaries that can be approximated within a given error and depth bound [66].

Reachability is dictated by circuit complexity and decoherence-limited depth. For example, with gate set $\{G_A, G_B\}$, the states reachable in one time step are $\{G_A, G_B\}$, in two time steps



(a) Haar random unitaries on $|0\rangle$ (b) Equispaced angles for U3 gate (c) Equispaced golden-ratio points

Figure 1. 1-qubit data sets of size 512.

$\{G_AG_A, G_AG_B, G_BG_A, G_BG_B\}$, and so on; a different set $\{G_C, G_D\}$ will typically reach a different region of Hilbert space at the same depth. Because the Hilbert space dimension grows much faster than the number of reachable sequences (and their bounded-error neighborhoods), gate sets diverge strongly in their reachability profile at fixed depth. This behavior is reminiscent of no-free-lunch theorems in optimization and learning [67, 68]. We note that dataset dependence is a fundamental feature of the problem rather than merely a limitation: the No Free Lunch Theorem and the ubiquity of universality in quantum gate sets imply that almost any randomly chosen gate set is universal with high probability [44], so the relative quality of gate sets is only meaningful with respect to a distribution of target unitaries. Crucially, our results in section 4.2.2 demonstrate that gate sets optimized on Haar-random unitary datasets *generalize* to a structurally different benchmark suite of 45 circuits from MQT Bench spanning 2–5 qubits of real-world quantum algorithms, while maintaining their fidelity and depth advantages over the standard $\{H, T, CX\}$ gate set, providing evidence that the discovered gate sets capture broadly useful decomposition properties rather than overfitting to the training distribution. For large unitaries, the dataset thus strongly shapes how YAQQ ranks gate sets and what 'optimal' means in practice.

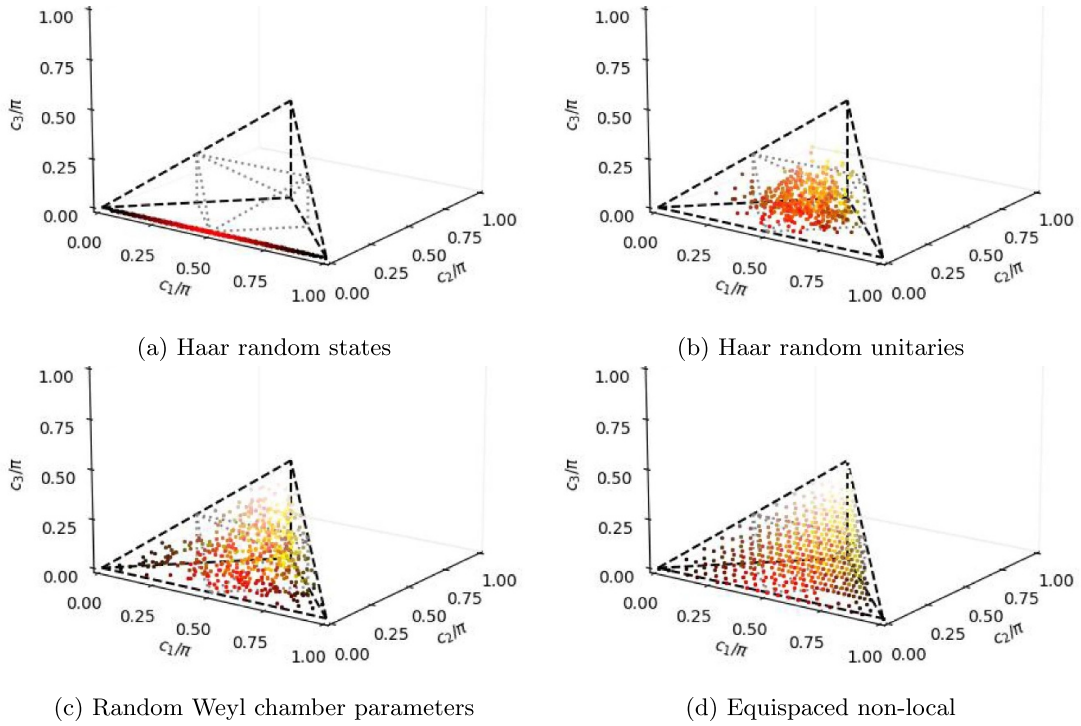
Within this framework, YAQQ provides several concrete dataset options. Datasets for 1- and 2-qubit unitaries can be visualized on the Bloch sphere and Weyl chamber, respectively. By default, YAQQ uses Haar-random unitaries of a user-specified sample size for an n-qubit Hilbert space. State-preparation unitaries for Haar-random states can also be selected; these are closely related, since a Haar-random unitary acting on any fixed state (e.g. $|0\rangle$) produces a Haar-random state [69]. This state–unitary relation differs from discrete classical [60] or discrete quantum circuit models [66], where 'random programs' induce a different distribution.

For 1-qubit datasets, YAQQ provides: (a) equispaced points on the Bloch sphere generated by golden-ratio rotations in spherical coordinates, (b) uniformly spaced parameters for IBM's U3 gate [70] (see equation (1)), and (c) stabilizer and magic states for evaluating transversal logical gate sets in QEC codes. For 2-qubit datasets, YAQQ provides: (a) equispaced non-local gates on the Weyl chamber, and (b) uniformly spaced Weyl-chamber coordinates. Figures 1 and 2 illustrate these 1- and 2-qubit dataset options. All datasets are internally stored as unitary matrices.

In addition to these built-in options, YAQQ can be easily configured with use-case-specific datasets, as demonstrated in section 4.2. This allows practitioners to tailor the notion of 'good' or 'novel' gate sets to particular algorithms, architectures, or error-correcting codes.

3.5. Decomposition techniques

To evaluate a candidate gate set, YAQQ must decompose each target unitary in the dataset into a circuit over that set. We support four decomposition techniques, which can be combined depending on the unitary size and the available gates: (i) SKD for 1-qubit unitaries to a discrete gate set, (ii) QSD for n-qubit unitaries into CX and single-qubit rotations, (iii) Cartan (KAK) decomposition for mapping between different 2-qubit entangling gates plus local rotations, and (iv) RD for direct stochastic synthesis using an arbitrary gate set. In practice, YAQQ primarily uses QSD, KAK, and RD; SKD is treated as a special case of 1-qubit decomposition and is included with these methods. The algorithms are detailed in appendix A.



(a) Haar random states

(b) Haar random unitaries

(c) Random Weyl chamber parameters

(d) Equispaced non-local

Figure 2. 2-qubit Data Sets of size 508. The axes corresponds to the t_x, t_y, t_z parameters in the 2-qubit canonical gate as defined in equation (2) in section 3.2.

3.6. Search technique

YAQQ can be used to compare two hard-coded gate sets based on the data set and decomposition method, as well as to decompose a unitary with a given gate set. This mode of operation can generate insights into already known gate sets or create a compiler on new data once a gate set is found. However, the main utility of YAQQ is in generating a novel complementary gate set based on the data set and decomposition method.

While the Hilbert space of the data set grows exponentially, the space of the gate set is also continuous. We consider various heuristics to intelligently explore this space. The search happens in two steps. First, a gate set is defined parametrically. The parameters are then adjusted based on the cost function.

In the second step, we use two techniques: either a stochastic search over the parameter space, which is denoted as RS or an optimization routine (e.g. SciPy), which encapsulates the cost function for a parametric gate set definition with a specific decomposition and data set. There are various options for global optimization (e.g. brute-force) or unconstrained minimization of multivariate scalar functions (e.g. Nelder-Mead, Powell, L-BFGS-B, COBYLA, SLSQP). Our experiments found that the COBYLA optimizer (CO) [71] achieves faster and better convergence. COBYLA's derivative-free trust-region method makes it particularly suitable for optimization landscapes where gradients are unavailable or computationally expensive. It thus performs well [72] for quantum program search, where the primary goal is often rapid exploration of the parameter space rather than precise convergence to the global optimum. In YAQQ, this option is denoted as CO($\vec{p}$), where $\vec{p}$ is the vector of parameters that define the optimization settings.

3.7. YAQQ software framework

Here we elaborate on the open-sourced `qiskit` implementation of the novelty search on quantum gate sets, i.e. YAQQ. It is implemented in Python and available on the Python Package Index (PyPI) [73]. The package is designed to be intuitive to install and use for researchers without significant software development background. For various mathematical operations and visualization, it depends on the Python libraries of `numpy`, `scipy`, `qutip`, `astropy`, `weylchamber`, `matplotlib`, and `tqdm`. The workflow and the organization of the package modules are detailed in appendix C.

Depending on the empirical framework defined above, in section 4 we conduct benchmarking studies with YAQQ with a random dataset (generated utilizing the process described in section 3.4). Further, in section 4.2.1 we investigate the applicability of the YAQQ framework and compare the performance of the YAQQ-generated gateset with the set of transversal logical gates, an important component of

quantum error correction (QEC) codes. Finally, in section 4.2.2, we show the scalability of YAQQ up to 5-qubits utilizing the MQT bench [14].

4. Results

In this section, we present some benchmarks selected to evaluate YAQQ. The sub-component configurations of these experiments are carefully designed to demonstrate specific features of YAQQ. These designs and corresponding results are presented in the following sub-sections.

4.1. Experiments with random datasets

We evaluate YAQQ on small random datasets of target unitaries, using the experiment grid in table 1 as our main configuration reference. Gateset GS1 is fixed to H1, T1 for 1-qubit targets and H1, T1, CX2 for 2 qubits and above, reflecting their status as the canonical FTQC-inspired native gate set; daggered variants are implicitly included whenever SKT-based decompositions are used. Each experiment in table 1 selects a candidate gateset GS2, a decomposition strategy for GS1 and GS2 (SKT, RND, KAK, QSD), a cost-function weight vector, and a search mode (SciPy-based CO or random search RS), thereby probing different aspects of YAQQ such as complementary gate-set discovery (experiments (ExpID) 1.1, 1.8), over-specification of universal sets (ExpID 1.2), random vs. structured decomposition (ExpID 1.3–1.4), hyperparameter tuning of the cost function (ExpID 1.5–1.6), gate-set comparison mode (ExpID 1.7), and 2-qubit extensions via SPE2 and NL2 (ExpID 2.1–2.3, ExpID 3.1).

For the numerical benchmarks that follow, we focus on these configurations as a compact design space for novel gate-set search on 1-, 2-, and 3-qubit Haar-random and algorithm-derived unitaries, always using the joint cost in equation (3) with weights chosen to emphasize process fidelity and, when indicated, novelty.

Concretely, from table 1 we use ExpID 1.1–1.8 to select the preferred cost weights and optimizer settings (CO with weight vector unless otherwise stated). ExpID 2.1–3.1 to validate multi-qubit decompositions via KAK and QSD while assessing how optimized GS2 gate sets trade off circuit depth and fidelity against the baseline GS1. Guided by this grid, the remainder of the section is organized into: (1) hyperparameter optimization of the cost-function and search method; (2) optimal benchmarking on 1-qubit unitaries against IBM-like native gate sets; and (3) scalability experiments on 2-qubit unitaries that quantify how these optimized novel gate sets extend to larger Hilbert spaces.

4.1.1. Hyperparameter optimization

Before applying YAQQ to automated discovery of novel gate sets, QEC, and unitary synthesis via MQT Bench [14], it is essential to first perform a systematic hyperparameter optimization. Extensive empirical studies across a wide range of models and benchmarks have already demonstrated that careful hyperparameter tuning is crucial for achieving strong predictive performance, often yielding substantial gains over untuned or default configurations [74–76]. Careful tuning of learning rates, exploration parameters, and model architectures ensures that YAQQ operates in a near-optimal regime, so that subsequent performance assessments reflect the true capabilities of the framework rather than artefacts of suboptimal hyperparameter choices.

The crucial hyperparameters responsible for enhancing the performance of YAQQ are the weights of the cost function (in equation (3)), the number of parameterized gates in the novel gate set, and the search method we utilize to optimize the parameters of these gates. Another relatively less crucial hyperparameter that can affect the performance of YAQQ is the choice of the decomposition technique. In the following, we will discuss each of these parameters individually and their corresponding optimal setting to reveal the highest potential of YAQQ. We use the experiments 1.1–1.8 (from table 1) to find the optimal hyperparameters.

Adjusting the weights w_{apf}, w_{npf}, w_{acd}, w_{ncd} and w_{agf} are the five weights that need to be adjusted in YAQQ to optimize the cost function (equation (3)) to find the optimal novel gate sets. For the sake of simplicity, we denote them as the list of weights where $W = [w_{\text{apf}}, w_{\text{npf}}, w_{\text{acd}}, w_{\text{ncd}}, w_{\text{agf}}]$. For the sake of further simplicity, we choose the list as $[w_{\text{apf}} \geqslant 1, 1, 1, 1, 0]$, which means we are encouraging the YAQQ to find a novel gate set that improves the average process fidelity compared to the novel gate set (in this case is {H1, T1, TD1}) while turning off the quantum engineering part (which is relevant while dealing with real quantum devices) by setting $w_{\text{agf}} = 0$.

In figure 3, we plot the difference of process fidelity averaged over 10, 1-qubit Haar, random unitaries using the novel gate set {H1, T1, $\text{P1}_{\vec{a}}$} and {H1, T1} with respect to the weight w_{apf}. We observe that

Table 1. The list of experiments for YAQQ on random-unitary datasets, showing for each experiment grouped by target-qubit dimension via the leading digit of the ExpID (i.e. with ExpID '*n.x*' we refer to the experiment number '*x*' with *n*-qubit) the candidate novel gate set (GS2), the decomposition pipelines used for the baseline GS1 and GS2 (SKT, RND, KAK, QSD), the cost-function weight vector $[w_{\text{apf}}, w_{\text{npf}}, w_{\text{acd}}, w_{\text{ncd}}, w_{\text{agf}}]$, and the search mode (CO: SciPy-based constrained optimization, RS: random search, Mode 2: gate-set comparison without parameter optimization). Together, these settings span complementary-gate discovery, over-specification of universal sets, random vs. structured decomposition, cost-weight tuning, and multi-qubit extensions used in the YAQQ benchmarks.

ExpID	GS2	Dcmp. GS1	Dcmp. GS2	Cost coeffs.	Search
1.1	$\{\text{P1}_{\vec{a}}, \text{P1}_{\vec{b}}\}$	SKT	SKT	[1,1,1,1,0]	CO
1.2	$\{\text{H1}, \text{T1}, \text{P1}_{\vec{a}}\}$	SKT	SKT	[1,1,1,1,0]	CO
1.3	$\{\text{P1}_{\vec{a}}, \text{P1}_{\vec{b}}\}$	RND	RND	[1,1,1,1,0]	CO
1.4	$\{\text{H1}, \text{T1}, \text{P1}_{\vec{a}}\}$	RND	RND	[1,1,1,1,0]	CO
1.5	$\{\text{P1}_{\vec{a}}, \text{P1}_{\vec{b}}\}$	RND	RND	[0.5,1,1,1,0]	CO
1.6	$\{\text{H1}, \text{T1}, \text{P1}_{\vec{a}}\}$	RND	RND	[0.5,1,1,1,0]	CO
1.7	{H1, T1}	SKT	RND	[0.5,1,1,1,0]	Mode 2
1.8	$\{\text{R1}_a, \text{R1}_b\}$	SKT	SKT	[1,1,1,1,0]	RS
2.1	$\{\text{R1}, \text{P1}_{\vec{a}}, \text{SPE2}_{t_y}\}$	RND, KAK	RND, KAK	[0.5,1,1,1,0]	CO
2.2	$\{\text{P1}_{\vec{a}}, \text{P1}_{\vec{b}}, \text{SPE2}_{t_y}\}$	RND, KAK	RND, KAK	[0.5,1,1,1,0]	CO
2.3	$\{\text{P1}_{\vec{a}}, \text{P1}_{\vec{b}}, \text{NL2}_{\vec{r}}\}$	RND, KAK	RND, KAK	[0.5,1,1,1,0]	CO
3.1	$\{\text{P1}_{\vec{a}}, \text{P1}_{\vec{b}}, \text{SPE2}_{t_y}\}$	SKT, KAK, QSD	SKT, KAK, QSD	[0.5,1,1,1,0]	CO

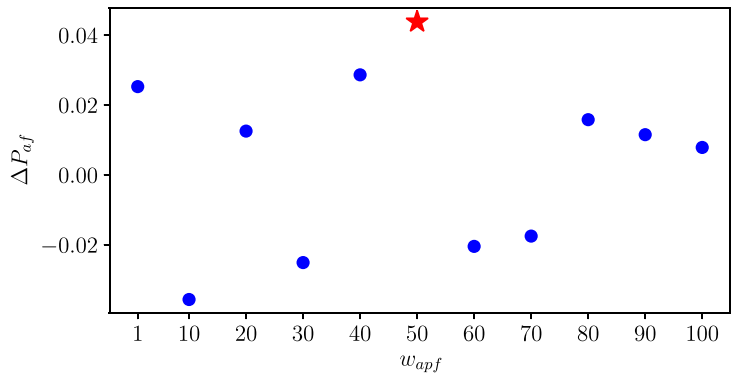


Figure 3. The optimal configuration of the cost function by adjusting the weights. The experiment is conducted on a dataset containing 1-qubit Haar random unitaries. The novel gate set contains {H1, T1} and a parameterized rotation $\text{P1}_{\vec{a}}$, optimized using the SciPy search. The performance of the optimized gate set is then compared with the primitive novel gate set {H1, T1}. Using these two gate sets, we decompose the dataset using the SKT. We investigate the variation in the difference in average process fidelity ($\Delta\langle\text{PF}\rangle$, where $\langle\text{PF}\rangle$ is the average process fidelity for a specific gate set) in decomposing 10 data points from the dataset with respect to the variation of w_{apf} while keeping the other weights fixed. The more positive the difference is, the more we improve the $\{\text{H1}, \text{T1}, \text{P1}_{\vec{a}}\}$ gate set compared to {H1, T1}. We observe the cost function performs optimally at $w_{\text{apf}} = 50$, indicating the optimal configuration of the cost function $[w_{\text{apf}}, w_{\text{npf}}, w_{\text{acd}}, w_{\text{ncd}}, w_{\text{agf}}] = [50,1,1,1,0]$.

the YAQQ performs optimally, indicating that the cost function is optimized when we set $w_{\text{apf}} = 50$ and keep the other weights fixed.

In figure 3, the weights [1,1,1,1,0] and [50,1,1,1,0] closely compete with each other in terms of providing us with the novel gate set that decomposes a Haar random unitary with very high fidelity. Hence, in table 2, we benchmark the performance of these two cost functions corresponding to the weights. As we emphasize average process fidelity ($\langle\text{PF}\rangle$) over average circuit depth ($\langle\text{CD}\rangle$), an improvement in PF for a deeper unitary decomposition is an acceptable trade-off. For a fixed search method and gate-set ansatz, we see that the cost function with the weight distribution [50,1,1,1,0] outperforms that with [1,1,1,1,0]. This investigation further shows that the optimal choice of weights for the cost function is [50,1,1,1,0] irrespective of the novel gate-set ansatz.

Preferred optimization method The two main optimization methods YAQQ utilizes are based on random search and SciPy optimizers, which are elaborately discussed in section 3.6. For the optimal setting of the cost function with weight distribution [50,1,1,1,0], in the right-hand side table of table 2 we can see that using random search method the novel gate set ansatz $\{\text{P1}_{\vec{a}}, \text{P1}_{\vec{b}}\}$ can achieve a higher fidelity compared to the native gate set {H1,T1} but the $\{\text{H1}, \text{T1}, \text{P1}_{\vec{a}}\}$ fails achieve it. Whereas when using the SciPy optimizer, we see that for $\{\text{H1}, \text{T1}, \text{P1}_{\vec{a}}\}$ the optimizer can find the optimal value of $\vec{a}$ that returns a higher fidelity in the decomposition task compared to the native gate set {H1, T1}. Interestingly, the

Table 2. Comparing the performance of the weight distribution [1,1,1,1,0] (left-hand side table) and [50,1,1,1,0] (right-hand side table) using two kinds of novel gate sets and search methods. We compare the primitive native gate set {H1, T1} with novel parameterized ansatz gate sets {H1, T1, $P1_{\vec{a}}$} and {$P1_{\vec{a}}$, $P1_{\vec{b}}$}. The YAQQ then uses either a random search or a SciPy optimizer to find the optimal gate parameters by optimizing $\vec{a}$ (and $\vec{b}$). From the weight distribution, it is evident that we are putting more stress on the average process fidelity (⟨PF⟩) than the average circuit depth (⟨CD⟩) of the decomposition achieved by novel gate sets, hence increasing the depth of decomposition for a higher fidelity is an acceptable trade-off. Hence, for a fixed optimization method, the [50,1,1,1,0] outperforms [1,1,1,1,0] in terms of providing us with a better average PF, further motivating the fact that the optimal cost function configuration can be achieved by fixing the weight [50,1,1,1,0].

Search	Gateset	⟨PF⟩	⟨CD⟩
Random	{H1, T1}	0.9432	255.45
	{$P1_{\vec{a}}$, $P1_{\vec{b}}$}	0.9331	**242.4**
	{H1, T1}	0.9389	284.7
	{H1, T1, $P1_{\vec{a}}$}	0.9366	**222.55**
SciPy	{H1, T1}	0.9167	258.65
	{$P1_{\vec{a}}$, $P1_{\vec{b}}$}	**0.9324**	**246.9**
	{H1, T1}	0.9426	271.3
	{H1, T1, $P1_{\vec{a}}$}	0.9339	303.9

Search	Gateset	⟨PF⟩	⟨CD⟩
Random	{H1, T1}	0.9161	239.4
	{$P1_{\vec{a}}$, $P1_{\vec{b}}$}	**0.9469**	269.2
	{H1, T1}	0.9310	221.65
	{H1, T1, $P1_{\vec{a}}$}	0.9193	264.65
SciPy	{H1, T1}	0.9285	260.55
	{$P1_{\vec{a}}$, $P1_{\vec{b}}$}	**0.9489**	**249.4**
	{H1, T1}	0.9305	218.7
	{H1, T1, $P1_{\vec{a}}$}	**0.9468**	337.5

Table 3. The performance of the Solovay–Kiteav (SKT) decomposition investigated over two novel gate set ansatz and search methods. Using the SKT decomposition, we observe that it yields a much shallower decomposition of unitaries than the random decomposition, regardless of the gate set or optimization method. Furthermore, we observe a notable decrease in average process fidelity (⟨PF⟩) using {H1, T1} gate set. But the same fidelity is achieved when the ansatz novel gate sets {$P1_a$, $P1_b$} and {H1, T1, $P1_a$} are used.

Search	Gateset	⟨PF⟩	⟨CD⟩
Random	{H1, T1}	0.7479	41.5
	{$P1_{\vec{a}}$, $P1_{\vec{b}}$}	**0.8131**	**5.05**
	{H1, T1}	0.6537	42.8
	{H1, T1, $P1_{\vec{a}}$}	**0.9166**	**21.05**
SciPy	{H1, T1}	0.7751	45.25
	{$P1_{\vec{a}}$, $P1_{\vec{b}}$}	**0.9268**	**5.75**
	{H1, T1}	0.7531	43.1
	{H1, T1, $P1_{\vec{a}}$}	**0.9217**	**13.6**

SciPy optimizer even enhances the performance of the {$P1_{\vec{a}}$, $P1_{\vec{b}}$} by providing us with a better average fidelity with a shallower depth decomposition, indicating the fact that the SciPy optimizer is preferable as a search method in comparison to random search.

The choice of decomposition In section 3.5 we discuss two ways to decompose a 1-qubit unitary. The first technique involves randomly decomposing the unitaries, whereas in the second case, we use the Solovay–Kiteav (SKT) decomposition. In table 3, we utilize the SKT decomposition instead of a random one. The investigation demonstrates that SKT decomposition is depth-efficient compared to a RD. Moreover, random search and the SciPy optimizers provide us with a higher average gate fidelity with nearly eight times less circuit depth for {$P1_a$, $P1_b$} ansatz gate set. Whereas in the case of {H1, T1, $P1_a$} we get a higher average fidelity, with nearly 2 (with random search) to 3 (with SciPy optimizers) times less circuit depth. This motivates us to use the SKT as the optimal decomposition for higher fidelity and a shorter depth unitary decomposition.

In summary, our initial investigation reveals that YAQQ achieves optimal performance under the following conditions: (1) setting the weights of the cost function (as defined in equation (3)) to [50,1,1,1,0], (2) optimizing the parameters of the ansatz gate set using the scipy COBYLA optimizer, and (3) decomposing the unitaries using the Solovay–Kitaev decomposition method.

Novel gate set design The benchmarking of YAQQ results primarily focuses on improving the average-case fidelity of the ansatz gate set, with little attention to the gate set's novelty. In this section, we further benchmark the performance of YAQQ by giving equal focus on the novelty of the ansatz gate set while decomposing 50, 1-qubit Haar random unitaries. The crucial factor that impacts the novelty of the

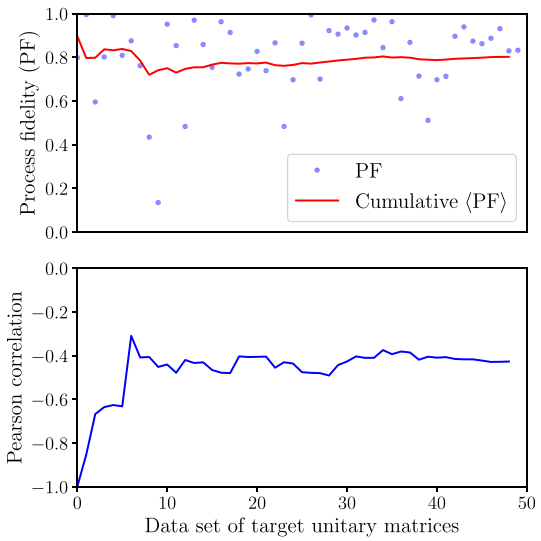


Figure 4. The novelty in the ansatz gate set $\{P1_{\vec{a}}, P1_{\vec{b}}\}$. The novelty is amplified by modifying the weight w_{npf} in the cost function. w_{npf} weights the novelty in the process fidelity, making the process fidelity (PF) achieved with the $\{P1_{\vec{a}}, P1_{\vec{b}}\}$ anti-correlated to the PF achieved with {H1, T1}. The anti-correlation in process fidelity increases as we consider more data but saturates close to −0.5, which still corresponds to a strong anti-correlation between the PF of ansatz and the novel gate set.

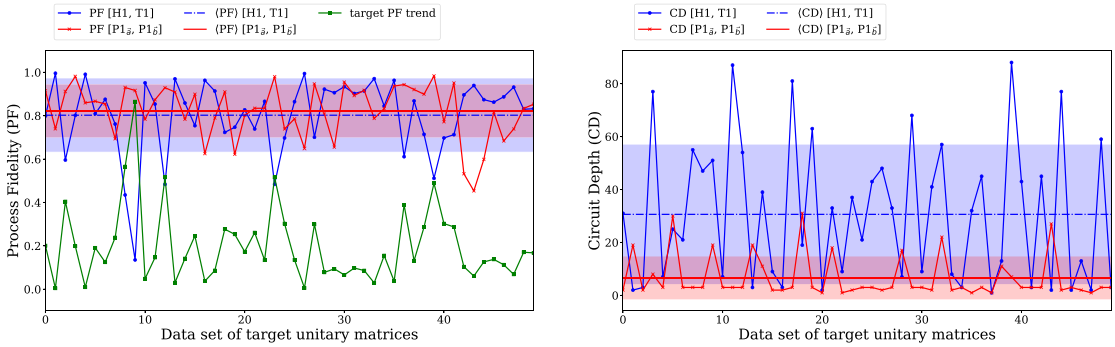


Figure 5. The process fidelity (PF) obtained by the optimized ansatz gate set is strongly anti-correlated to the PF of the novel gate set. This indicates the novelty in the ansatz gate set. In this task, we use 50 instances of 1-qubit Haar random unitaries, which are decomposed using a novel gate set {H1, T1} and ansatz gate set $\{P1_{\vec{a}}, P1_{\vec{b}}\}$ using SKT decomposition. The YAQQ optimizes the ansatz gate set using 500 iterations of `COBYLA` optimizer. Moreover, the ⟨PF⟩ for the novel gate set is higher than the original.

ansatz gate set is choosing the cost function weights carefully. As the quantifier of the novelty, we utilize Pearson's correlation function (P_{crr}), which varies between −1 and 1 where if $-1 \leqslant P_{\text{crr}} < 0$, then it defines anti-correlation and $0 > P_{\text{crr}} \geqslant 1$ corresponds correlation between two given dataset. Novelty in the gate set corresponds to an anti-correlation between the process fidelity obtained through the ansatz gate set and the novel gate set. For this experiment, $\{P1_{\vec{a}}, P1_{\vec{b}}\}$ were chosen as the ansatz gate set and {H1, T1} as the novel gate set.

We recall that the weight w_{npf} in the cost function impacts the novelty of the ansatz gate set. In figure 4, we set $w_{\text{npf}} = 20$, which modifies the optimal setting of the weights to [50,20,1,1,0], and we can see that the average Pearson correlation of the process fidelity obtained by the gate sets strongly anti-correlates (i.e. $P_{\text{crr}} \leqslant 0.5$) for 10 Haar random unitaries. Afterward, as we increase the number of samples in unitary, the anti-correlation saturates close to −0.5.

In figure 5, we plot the process fidelity of decomposing 50 random unitaries for the novel and the ansatz gate set with the weight distribution of the cost function [50,20,1,1,0]. The results show that the trend in the variation of the process fidelity in the ansatz gate set is anti-correlated to the process fidelity obtained through the novel gate set. Moreover, manipulating the cost function induces strong novelty in the optimized ansatz gate set, yet the average fidelity also increases compared to the novel gate set.

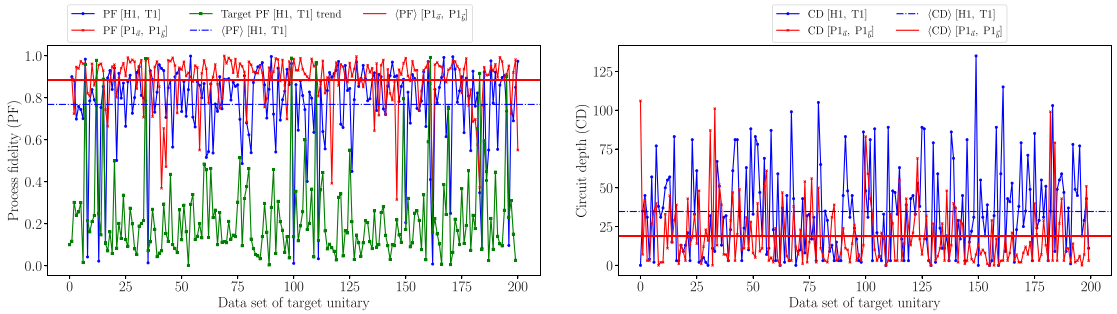


Figure 6. The ansatz gate set, we can decompose the unitaries with higher average fidelity than the average of the novel gate set. The results are gathered while decomposing 200 Haar random unitaries using the novel gate set {H1, T1} and the ansatz gate set {P1$_a$, P1$_b$}. Moreover, the decomposition depth of unitaries is shallower using the ansatz gate set than the novel gate set.

After optimizing the ansatz gate set {P1$_{\vec{a}}$, P1$_{\vec{b}}$} using 500 iterations of `COBYLA` optimizer we obtain the novel gate set as follows:

$$\mathrm{P1}_{\vec{a}}^{\mathrm{nov.50}} = \begin{bmatrix} +0.95695777 + 0.j & -0.25428694 - 0.13989277j \\ +0.20772632 + 0.20268599j & +0.27797849 + 0.91569434j \end{bmatrix},$$

$$\mathrm{P1}_{\vec{b}}^{\mathrm{nov.50}} = \begin{bmatrix} -0.23161363 + 0.j & -0.97275633 + 0.01001202j \\ +0.96339857 - 0.13497524j & -0.22903051 + 0.03449493j \end{bmatrix}.$$

4.1.2. Benchmarking gate set optimality

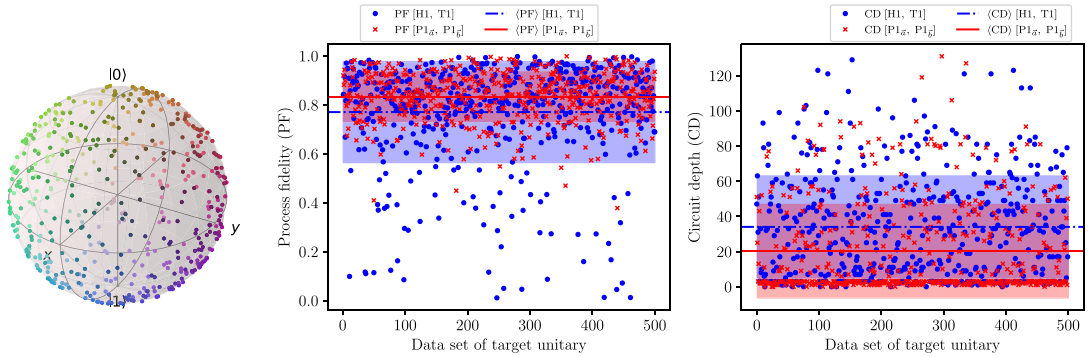
Utilizing the optimal setting from the previous section, we benchmark the YAQQ's performance in designing a gate set that provides an optimal decomposition to Haar random unitaries. Throughout the experiments, we consider {P1$_{\vec{a}}$, P1$_{\vec{b}}$} as the ansatz gate set where the parameters $\vec{a}$ and $\vec{b}$ are optimized using 1000 iterations of COBYLA optimizer (CO).

In figure 6, we consider 200 Haar random unitaries where we show that the average process fidelity and the depth with the ansatz gate set outperforms the novel gate set. In the optimal configuration of the cost function, the weight w_{npf}, which is responsible for deciding the novelty of the YAQQ, is set to 1. That is why even though we get an increase in average process fidelity with a shallower decomposition using the ansatz gate set, the gate set lacks novelty. Hence, motivated by this observation, in the next experiment, we modify the weights of the optimal cost function by turning w_{npf} as a variable. So, the weights of the cost function become [50,w_{npf},1,1,0]. As the w_{npf} increases, we expect to observe an increase in anti-correlation between the trend in average process fidelity obtained using the novel and the ansatz gate set. To measure anti-correlation, we utilize Pearson's correlation coefficient [77].

$$\mathrm{P1}_{\vec{a}}^{\mathrm{opt.200}} = \begin{bmatrix} -0.78687181 + 0.j & -0.04577325 - 0.61541657j \\ +0.03316088 - 0.61622488j & -0.7867068 + 0.01611391j \end{bmatrix}$$

$$\mathrm{P1}_{\vec{b}}^{\mathrm{opt.200}} = \begin{bmatrix} +0.8915669 + 0.j & +0.22308591 - 0.39413341j \\ +0.24090997 - 0.38349818j & +0.42340431 + 0.78461476j \end{bmatrix}$$

In figure 7, we then extend the m unitaries and show that the YAQQ can fine-tune the {P1, P1} gate set for fidelity. The gates found are:



(a) Visualization of 500 Haar random unitaries.

(b) The comparison of the {H1, T1} and the ansatz gate set {P1$_a$, P1$_b$}.

Figure 7. Comparison of the performance of YAQQ in decomposing 500 Haar random unitaries with {H1, T1} and the ansatz gate set {P1$_{\vec{a}}$, P1$_{\vec{b}}$}. In this experiment, we choose the optimal setting of the YAQQ where the Haar random unitaries are decomposed using SKT decomposition, the weights of the cost function are set to [50,1,1,1,0], and the parameters of the ansatz gate set are optimized using SciPy COBYLA optimizer. As the performance quantifier, we investigate the process fidelity (PF) and the circuit depth (CD). In figure (b), we can see that the ⟨PF⟩ of decomposing 500 unitaries with the novel ansatz {P1$_{\vec{a}}$, P1$_{\vec{b}}$} is more accurate compared to the {H1, T1} gate set. Furthermore, the ansatz gate set accomplishes this task using a shallower circuit.

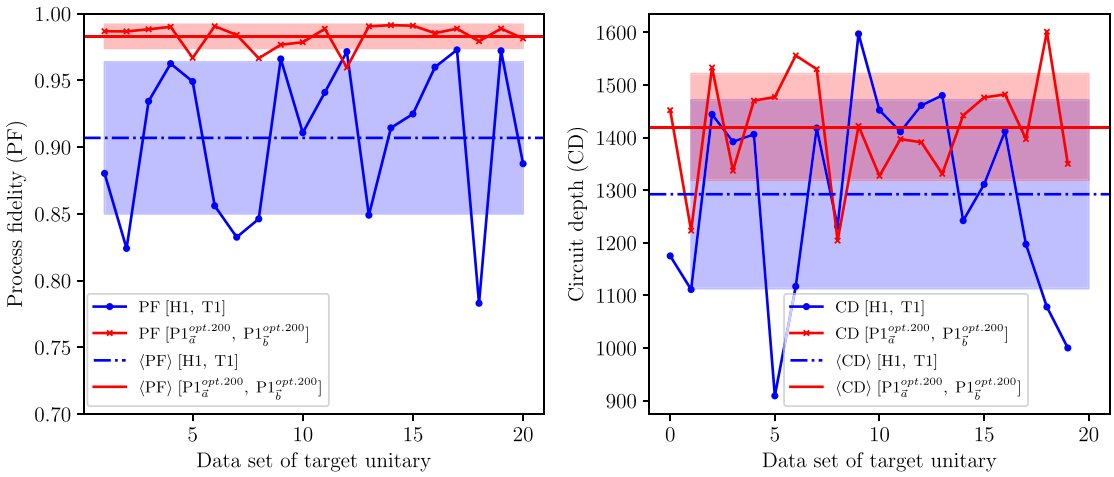


Figure 8. The original and optimal novel gate sets from the experiment presented in figure 6 are extended with the CX2 gate. This allows the decomposing of n-qubit unitaries. Here, the performance of 20 Haar random 2-qubit unitary matrices is decomposed using the KAK and SKT. This demonstrates the scalability of the YAQQ by generalizing results to a larger Hilbert space.

$$\mathrm{P1}_{\vec{a}}^{\mathrm{opt.500}} = \begin{bmatrix} -0.99891178 + 0.j & -0.01683013 - 0.04349723j \\ -0.0406026 + 0.02294975j & +0.7722138 + 0.63364862j \end{bmatrix}$$

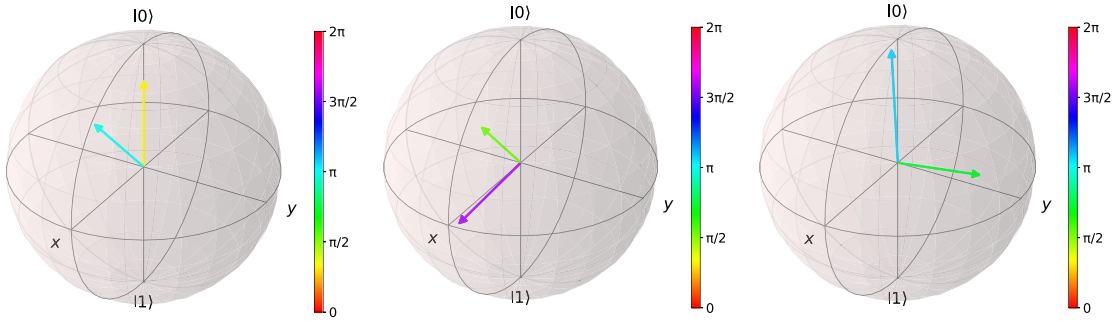
$$\mathrm{P1}_{\vec{b}}^{\mathrm{opt.500}} = \begin{bmatrix} +0.54683177 + 0.j & -0.52979855 - 0.64829662j \\ +0.8231603 + 0.15291220j & +0.26287606 + 0.47950095j \end{bmatrix}.$$

4.1.3. Optimal scalability

The optimal gate set on one qubit data set can now be augmented to a general 2-qubit gate set by adding a special perfect entangler. We extend both the original gate set {H1, T1} and {P1$_{\vec{a}}^{\mathrm{opt.200}}$, P1$_{\vec{b}}^{\mathrm{opt.200}}$} gate set by adding CX2 as the canonical entangling gate. Data on extensive set of experiments with SPE2 and NL2, as referred to in table 1 can be accessed from YAQQ's public repository. The comparison of the performance of these two gate sets is shown in figure 8. For this experiment, the optimal P1 gates are supplied to YAQQ via the F1 (load gate from file) option, based on the saved gates from the previous experiment of figure 6. We demonstrated the scalability of the novel gate set for multi-qubit decomposition. The two-qubit decomposition uses the KAK decomposition. Similarly, this gate set can also be used to decompose *n*-qubit unitaries via the QSD option.

4.1.4. An interpretation of the discovered gate sets

In this section we discuss the physical interpretation of the discovered gate sets. We do this by visualizing the rotations on the Bloch sphere as depicted in figure 9. We observe that the elements of the gate



(a) H and T (b) $\mathrm{P1}_{\vec{a}}^{opt.200}$ and $\mathrm{P1}_{\vec{b}}^{opt.200}$ (c) $\mathrm{P1}_{\vec{a}}^{opt.500}$ and $\mathrm{P1}_{\vec{b}}^{opt.500}$

Figure 9. Visualization of single qubit gate sets on the Bloch sphere. Each gate, $U \in SU(2)$ is represented as $U = \exp\left(-\mathrm{i}\frac{\alpha}{2}\hat{n}\cdot\hat{\sigma}\right) = \cos\left(\frac{\alpha}{2}\right)I - \mathrm{i}\sin\left(\frac{\alpha}{2}\right)(\hat{n}\cdot\hat{\sigma})$, where, $\alpha \in [0, 2\pi)$ is the angle of rotation around $\hat{n} = (n_x, n_y, n_z)$ axis, and $\hat{\sigma} = (\sigma_x, \sigma_y, \sigma_z)$ are the Pauli matrices.

sets exhibit rotation about two distinct, non-orthogonal axes. This is desirable from a group theoretic perspective as efficient single qubit synthesis requires operations that can move a state along different trajectories.

Moreover, we note that typically one of the gates closely corresponds to a cardinal direction based on platonic solid group generators while the other is T-like, i.e. acts transitively on the edges. This allows for efficient covering of $SU(2)$ inline with super-golden gate constructions [16] based on S-arithmetic unit quaternion groups.

4.2. Experiments with application-specific datasets

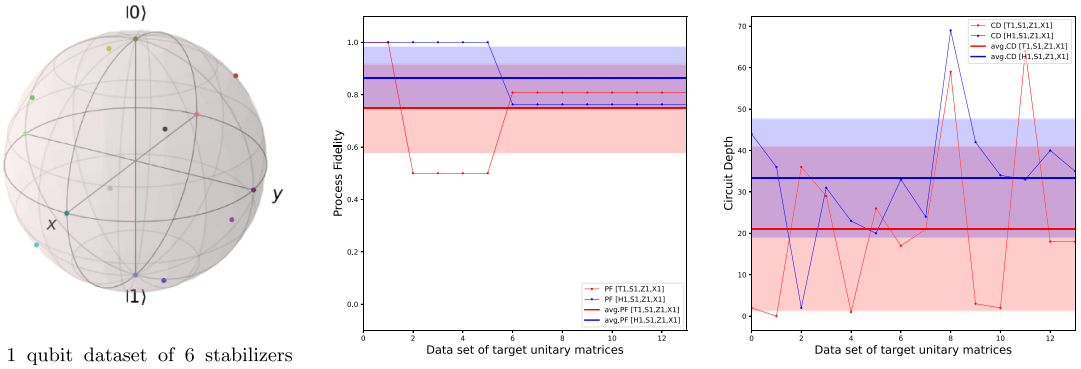
This section presents two exemplary applications where YAQQ can be applied. We also discuss some related works and approaches used in other toolsets for these applications.

4.2.1. Comparison of QEC codes

Transversality is an important feature of FTQC. Transversal logical gates are those for which an error-correcting code can achieve a transformation on a logical qubit by applying that gate to each of the physical qubits [78]. For example, in a 7-qubit Steane code, logical Hadamard can be performed by applying Hadamard on each of the 7 physical qubits This feature makes these gates the simplest to implement because they are automatically fault-tolerant, i.e. if an error occurs on one physical qubit, no action can propagate it onto a different physical qubit in the same code because no two qubits in the block every interact. These logical gates are typically also short gate sequences, making the corresponding fault-tolerant threshold particularly high. However, the Eastin–Knill theorem [50] states that a universal gate set cannot be transversal in any QEC code. Comparing transversal logical gates to the fault-tolerance capabilities of a QEC code is an active research field [79] that can benefit from empirical experiments on YAQQ.

In an experiment to compare transversal logical gate sets, we used the YAQQ's compare functionality. We specifically compared the Reed-Mueller code, having the transversal gate set {T1, X1, S1, Z1, CZ2}, with the Steane code, having the transversal gate set {H1, X1, S1, Z1, CX2} [80]. The dataset was specifically designed for the 1-qubit quantum states composed of 6 stabilizers and 8 magic states, as shown in figure 10(a). These are typically considered the easiest and hardest states to represent in QEC. However, as a tradeoff, codes with easy preparation of Clifford states (e.g. in Steane or CSS codes) do not have the easy preparation of magic gates like T1 (e.g. in Reed-Mueller). This tradeoff is shown in figure 10(b), where the fidelity of magic states is better with the Reed-Mueller (in red) code, while the fidelity of cardinal states in the X and Y direction is difficult without an H1 gate. Conversely, the stabilizer states are trivial for the Steane code, while they fail for the magic states. Based on experiments, we found the Reed-Mueller code to be more amenable to the Solovay–Kitaev decomposition, and conversely, the Steane code performed better for RD. The raw experiment data is available on the YAQQ Git repo.

As a generalization of the QEC application, YAQQ can be used to empirically demonstrate weak universality by showing that all gates of a known universal gate set have a bounded depth bounded fidelity decomposition in the new gate set.



(a) 1 qubit dataset of 6 stabilizers and 8 magic states

(b) Comparison result

Figure 10. Comparison of transversal gate sets in quantum Reed-Muller and Steane codes. We infer that the fidelities of magic states (data points $6-13$) are better with the Reed-Mueller code (in red), while the fidelities of cardinal states (data points $0-5$) in the X and Y direction are difficult without an H1 gate. Conversely, the stabilizer states are trivial for the Steane code (in blue) while failing to reach the magic states.

4.2.2. Designing optimal quantum instruction sets

YAQQ challenges the canonical gate set used in QC for efficient circuit decomposition. Though most quantum processors allow generic rotation gates, from the perspective of quantum control, only a discrete gate set can be assigned a finite number of representations. These can be at either of the 3 levels of (i) quantum assembly, (ii) QISA, or (iii) quantum pulses. Thus, with YAQQ, we can determine which settings of the rotation angles (of, say, the generic P1 gate) can be hardcoded as a specific quantum instruction. This is an important problem in the design of the quantum microarchitectures [81]. Similar to how the {H1, T1} 1 qubit universality is proven by the SKT, and we understand their importance for entanglement generation and magic-state, the novel gate sets might lead us to discover new properties [82] of quantum information processing (e.g. super golden gates [16, 83] and special perfect entanglers [84, 85]) and aid in the understanding [86] of quantum resources [59]. Specifically, it would be crucial to explore the universal distribution [66] of quantum states from these gate sets and how the results compare with circuit [87] and algorithmic information theory [88, 89].

On the other hand, using an over-specified gate set, we can discover new concepts composed of the original gate set; for example, H followed by CX is often used to entangle qubits and thus can be a useful gate that reduces the circuit depth by 1 for every usage. Such exploration has been done before in [58]. In [90], we use YAQQ to optimize the QISA encoding, inspired by similar work in classical assembly targeted for low-power embedded systems. This set of gates (or quantum instructions) optimizes the quantum instruction bandwidth and energy budget between the quantum processor and quantum compiler. As a proof-of-concept for this work, we used the optimal gate sets based on 200 Haar random unitaries to decompose 45 unitaries of well-known quantum algorithms from the MQT Bench [14]. The comparative process fidelity and circuit depth of using the $\{\mathrm{P1}_{\vec{a}}^{\mathrm{opt.200}}, \mathrm{P1}_{\vec{b}}^{\mathrm{opt.200}}\}$ gate set with respect to the standard {H1, T1}, (with both gate sets augments with the CX2 gate) for decomposition is shown in figure 11. We see similar trends as the 1- and 2-qubit random unitary experiments discussed earlier in figures 6 and 8. This confirms that the dataset results from random dataset experiments of YAQQ can be utilized in known usage of quantum processors. Moreover, it confirms that the $\{\mathrm{P1}_{\vec{a}}^{\mathrm{opt.200}}, \mathrm{P1}_{\vec{b}}^{\mathrm{opt.200}}, \mathrm{CX2}\}$ gate set performs surprisingly better than the standard {H1, T1, CX2} gate set in decomposition fidelity, while maintaining a slightly higher circuit depth. Thus, this gate set can be further studied for QEC and pulse-level specification.

QAS is used widely in variational quantum algorithms for ansatz circuit design [91, 92]. These ansatz can be defined as custom parametric gates for comparison. Similarly, quantum circuit learning [93] directly infers a quantum circuit for a required use case. By incorporating similar concept discovery and decompilation techniques, YAQQ can be used to aid the design of quantum algorithms and ansatz [65, 94–97] in the future.

4.3. Discussion on noise sensitivity and hardware applicability

Throughout the article the experiments are conducted in a noise-free simulation environment, hence in this section we discuss how the results are connect to a realistic hardware settings. Within the present scope, circuit depth serves as a coarse proxy for noise sensitivity. Deeper circuits are more susceptible to decoherence and accumulate longer gate execution times, both of which degrade output fidelity on

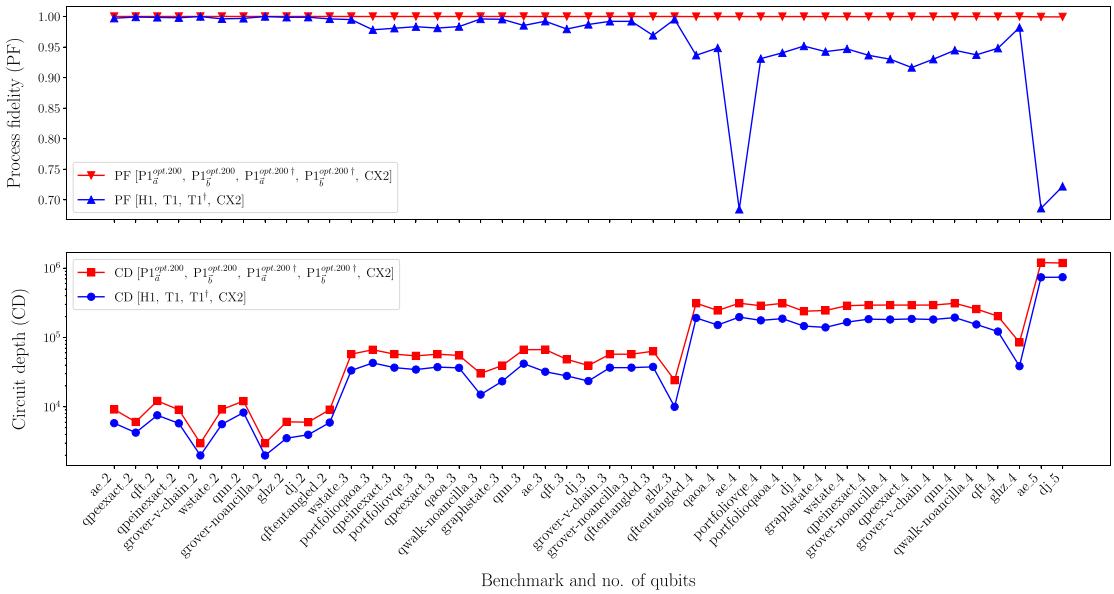


Figure 11. Process fidelity and circuit depth of {P1$_{\vec{a}}^{\text{opt.200}}$, P1$_{\vec{b}}^{\text{opt.200}}$, CX2} vs {H1, T1, CX2} for 45 unitaries of real-world quantum algorithm from MQT Bench.

physical hardware [98]. Concretely, the joint cost function in equation (3) simultaneously penalizes decomposition infidelity and circuit depth, so that both principal error sources at the decomposition stage of transpilation, the approximation error intrinsic to finite gate-set decomposition and the expected decoherence-induced infidelity implied by circuit depth, are at least indirectly accounted for, even in the absence of an explicit noise model.

For the NISQ-era setting, this means that a gate set discovered by YAQQ to have lower average circuit depth and higher decomposition fidelity is expected to perform better on noisy hardware, since it reduces both error contributions simultaneously. A more precise characterization would require incorporating device-specific gate error rates and coherence times into the c_{agf} term of the cost function in equation (3), which constitutes an important direction for future work.

For the FTQC-relevant setting, the connection to hardware noise takes a different form. YAQQ targets the *logical decomposition layer* [99], where algorithms are compiled into a discrete set of transversal logical gates for a QEC code. At this layer, the underlying physical noise is already suppressed by the QEC cycle, and the dominant resource cost shifts to the non-transversal gates, most prominently non-Clifford gates such as the T gate, whose fault-tolerant implementation requires expensive magic-state distillation. Consequently, our noise-free circuit analysis is directly applicable in the FTQC setting: minimizing circuit depth and gate count over the logical gate set directly reduces magic-state distillation overhead and logical error rates, making the YAQQ optimization criterion well-aligned with the dominant cost model for fault-tolerant QC. The transversal gate set experiments in section 4.2.1 demonstrate this use case concretely.

5. Conclusion

This article introduces the tool YAQQ. It enables the comparison of quantum gate sets by decomposing a set of specified quantum unitaries. YAQQ can be used as a generic quantum compiler, as well as for benchmarking quantum processors based on their native gate sets. More importantly, this comparison capability allows YAQQ to perform DSEs of quantum gate sets. A novel gate set can be specified as a collection of fixed, parametric, or random gates, which YAQQ can thereafter optimize with respect to another specified gate set.

The formal rationale underlying YAQQ is grounded in algorithmic information theory and quantum universality. It thus allows for comparing universal and sub-universal computational models with similar theoretical expressibility but different practical reachability. This subjective evaluation is based on the set of target algorithms or the required quantum transformations for a specific use case and between two gate sets.

In this article, after detailing the various steps involved in this process, we presented experiments demonstrating some insights about quantum gate sets and circuit compilation. We found 2 sets of gate sets, based on 200 and 500 random unitaries, respectively, which perform better than conventional gate sets on the tuned multi-modal cost function comprising of process fidelity of the approximate decomposition, the quantum circuit depth, and a novelty score. These performance results were shown to generalize to larger unitaries and real-world quantum algorithms, proving the efficacy of the YAQQ DSE method. Regarding scalability, YAQQ's gate-set search space is intentionally restricted to 1- and 2-qubit gates, since universality can be achieved with one 2-qubit entangling gate, one Clifford gate, and one non-Clifford gate at minimum, without requiring higher-order gates. Within this space, we demonstrated circuit-level scalability up to 3-qubit unitaries via QSD and validated the discovered gate sets on 45 real-world algorithm circuits spanning 2–5 qubits from the MQT Bench suite, confirming that the optimized gate sets generalize beyond the random-unitary training distribution. Thereafter, a few exemplary use cases of YAQQ are presented. This includes comparing resources for different QEC codes, optimizing quantum instruction sets for a set of quantum algorithms, and optimizing quantum control at the pulse level. With respect to hardware applicability, we emphasize that in a FTQC setting the relevant target gate set at the decomposition layer refers to the logical gate sets of QEC codes rather than the underlying physical gates used within the QEC cycle. The transversal gate set experiments in section 4.2.1 demonstrate this use case directly. For NISQ-era applicability, YAQQ uses circuit depth as a coarse proxy for noise sensitivity, since deeper circuits are more susceptible to decoherence; the joint cost function in equation (3) therefore simultaneously penalizes both the approximation infidelity intrinsic to finite gate-set decomposition and the expected decoherence-related infidelity implied by circuit depth, so that both principal error sources at the decomposition stage of transpilation are at least indirectly accounted for. Incorporating explicit noise models and device-specific error rates into the cost function via the c_{agf} term constitutes an important direction for future work. These applications will be more thoroughly studied in our future work. Regarding the role of novelty search specifically, the observed fidelity gains in the fidelity-driven experiments are primarily attributable to parameterized gate optimization (with the novelty weight intentionally suppressed at $w_{\mathrm{npf}}=1$ against $w_{\mathrm{apf}}=50$), while the isolated effect of novelty is evidenced by the Pearson anti-correlation of ≈ -0.5 between baseline and discovered gate-set fidelity profiles when w_{npf} is increased (section 4.1.1), confirming that the novelty component induces a qualitatively distinct behavioral effect beyond pure average-fidelity optimization. A controlled ablation comparing a novelty-free cost function against the full YAQQ objective at identical gate-set ansatz constitutes an important direction for future work. Furthermore, the time-consuming DSE within YAQQ can be accelerated on GPUs or by using deep reinforcement learning-based models. In light of the surveyed functionalities, we expect the open-sourced YAQQ package to become an indispensable tool for quantum computer architects both in the resource-constrained NISQ era and the discrete-gate-set-constrained FTQC era.

Acknowledgments

AS thanks Pablo le Henaff for insightful discussions regarding Cartan decompositions and Zaid Al-Ars for discussions on using YAQQ as a compiler for NISQ processors. AS acknowledges funding from the Dutch Research Council (NWO) through the project 'QuTech Part III Application-based research' (Project No. 601.QT.001 Part III-C—NISQ). AK and JM were partially supported by the Polish National Science Center (NCN) under the Grant Agreement 2019/33/B/ST6/02011. MS and SF are grateful for financial support from Intel.

Software availability

YAQQ is available on the Python Package Index (PyPI) [73]. The open-sourced code for YAQQ, configuration files, output data, and plotting codes for the experiments presented in this article are available at: https://github.com/Advanced-Research-Centre/YAQQ [100].

Author contributions

Conceptualization, A.S.; Methodology, A.S., A.K. and T.A.; Software, A.S. and A.K.; Validation, A.K., A.S., S.M., M.S. and S.Fa.; Investigation, A.S., A.K., S.M. and M.S.; Writing—Original Draft Preparation,

A.S., A.K. and M.S.; Writing—Review & Editing, M.S., J.M., S.Fe., A.S., A.K., T.A.; Visualization, A.K., A.S., T.A, S.M. and S.Fa.; Supervision, S.Fe., J.M. and A.S.; Project Administration, A.S. and A.K.; Funding Acquisition, S.Fe. and J.M.

Appendix A. Review of decomposition algorithms

Here we review additional details of the decomposition methods presented in section 3.5.

A.1. QSD

QSD was proposed [48, 101] as a technique for expressing any n-qubit quantum operator as an exact decomposition into single-qubit rotations and 2-qubit controlled gates. The algorithm follows a divide-and-conquer strategy, recursively breaking the n-qubit unitary matrix U into smaller submatrices. QSD starts with cosine-sine decomposition (CSD), a well-known linear-algebraic technique that partitions the target matrix U into smaller blocks. It then recursively applies CSD [102] and other decomposition techniques, such as eigenvalue and Euler decompositions, to ultimately express the original complex operator as a sequence of single-qubit gates and CNOTs. This synthesis technique is a quantum version of the classical Shannon decomposition of Boolean functions. The single-qubit gates and CNOT from the QSD can be further decomposed via SKT or RD to the available gate set.

According to CSD, $U = LMR^{\dagger}$ where L and R are block-diagonal matrices representing uniformly controlled gates and the middle matrix M representing a controlled R_y rotation on the MSB.

$$U = \left[\begin{array}{c|c} U_{00} & U_{01} \\ \hline U_{10} & U_{11} \end{array}\right] = \left[\begin{array}{c|c} L_1 & 0 \\ \hline 0 & L_2 \end{array}\right] \left[\begin{array}{c|c} C & -S \\ \hline S & C \end{array}\right] \left[\begin{array}{c|c} R_1 & 0 \\ \hline 0 & R_2 \end{array}\right]^{\dagger} \tag{4}$$

L_1, L_2, R_1 and R_2 are unitary matrices of size 2^{n-1}. C and S are diagonal matrices such that $C^2 + S^2 = I$, thereby justifying the name of the decomposition technique. The matrices L and R are termed as quantum multiplexors, and they enact L_1 (R_1) or L_2 (R_2) conditioned on the state of the MSB. The middle matrix resembles the R_y rotation matrix that is targeted on the MSB and controlled by the states of the lower-order qubits.

The left and right gates undergo a demultiplexing routine that performs an eigenvalue decomposition of the matrices.

$$\begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} = \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix} \begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda^{\dagger} \end{bmatrix} \begin{bmatrix} Q & 0 \\ 0 & Q \end{bmatrix} \tag{5}$$

P and Q are unitary matrices, and Λ is a unitary diagonal matrix. The left and right matrices are quantum gates operating on the lower-order qubits and are independent of the MSB. The middle matrix corresponds to a R_z operation on the MSB controlled by the lower qubits.

This process of CSD, followed by subsequent demultiplexing operation (see figure 13), is performed recursively until the algorithm reaches the base case. At the base level, the operator sequence consists of only single qubit gates. At this point, any single-qubit unitary operation is changed into a rotation gate following Euler decomposition. The implementation of the algorithm can have two options, $a1$ and $a2$. In $a1$, the multiplexed R_y operation in figure 12 is implemented using CZ gates, while in $a2$, the recursion is stopped at the level of 2-qubit operations, and the resulting circuit is decomposed into CNOT gates and single-qubit rotation gates.

In this project, the Qiskit implementation of the QSD algorithm is employed with the $a2$ option. It returns a decomposed quantum circuit consisting of single-qubit unitary rotations and CNOT gates as depicted in figure 14 for a 2-qubit system.

A.2. Canonical decomposition

The QSD decomposes n-qubit unitaries to CNOT and single-qubit rotations. The single-qubit rotations can further be decomposed into the target gate set via SKD. However, the target gate set might not include CNOT. This constraint necessitates the use of the Cartan decomposition (KAK) [103]. The acronym KAK [104] refers to the use of a group $\mathcal{G} = \exp(g)$ with a subgroup $\mathcal{K} = \exp(k)$ and a Cartan subalgebra a, where $g = k \oplus k^{\perp}$ and $a \subset k^{\perp}$. Any $G \in \mathcal{G}$ can be expressed as $G = K_1 A K_2$, where $K_1, K_2 \in \mathcal{K}$, and $A \in \exp(a)$. This special case of Cartan Decomposition allows one to factor a general 2-qubit operation (i.e. an element of $U(4)$) into local operations applied before and after a 3-parameter,

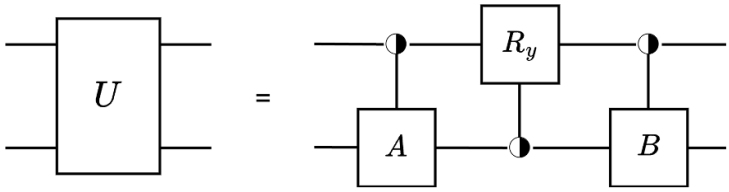


Figure 12. The Cosine-Sine decomposition acting on n-qubit gate U. The slash represents a bundle of wires, and the box control symbol indicates multiple control wires.

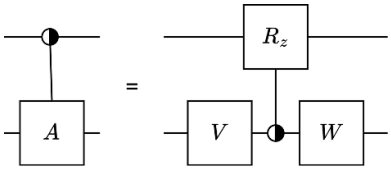


Figure 13. Demultiplexing of a multiplexor.

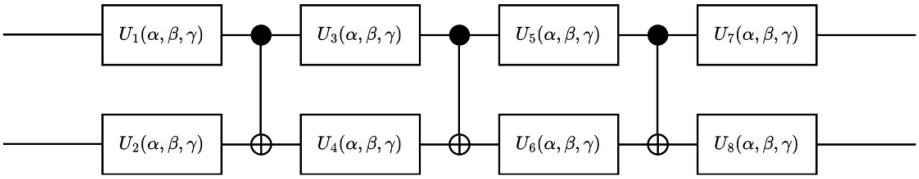


Figure 14. Final decomposed form of a 2-qubit arbitrary unitary operator expressed as a sequence of 1-q rotations and CNOT gates.

non-local operation. A general 2-qubit gate corresponds to a 4×4 unitary matrix with 16 free parameters (15 considering an irrelevant phase). Based on KAK decomposition, any 2-qubit gate can be expressed as a canonical gate, plus 4 local 1-qubit gates, thus tuning the $15 = 3 + 4 \times 3$ parameters. The canonical gate is defined in section 3.2. This decomposition based on the canonical gate is known as the magic- or Kraus-Cirac- decomposition [105]. Note that if the gate set provided does not have a perfect two-qubit entangler, the Qiskit implementation of approximate KAK decomposition would still return a circuit with the highest fidelity, though it could be rather low.

A general 2-qubit gate corresponds to a 4×4 unitary matrix, with 16 free parameters, i.e. 15 parameters and an irrelevant global phase. Fortunately, any 2-qubit gate can be decomposed into a 3-parameter canonical gate, plus 4 local 3-parameter (rotation angles across 3 axes) 1-qubit gates. Decomposition to the canonical gate is also known as the magic-, Kraus-Cirac-, or KAK-decomposition. This decomposes to the canonical representation, involving a similarity transformation to the magic basis.

$$V = M\,U\,M^{\dagger}$$

where M is the magic gate,

$$M = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & i & 0 & 0 \\ 0 & 0 & i & 1 \\ 0 & 0 & i & -1 \\ 1 & -i & 0 & 0 \end{bmatrix}$$

M diagonalizes the canonical gate as:

$$\mathrm{NL2}_{\vec{t}} = \mathrm{Can}\left(t_x, t_y, t_z\right) = M\,D\,M^{\dagger}$$

and, $D = \mathrm{diag}(e^{i\frac{1}{2}(+t_x-t_y+t_z)}, e^{i\frac{1}{2}(-t_x+t_y+t_z)}, e^{i\frac{1}{2}(+t_x+t_y-t_z)}, e^{i\frac{1}{2}(-t_x-t_y-t_z)})$.

If U is a special orthogonal matrix (i.e, Real, $U^T = U$, and $\det U = 1$), then in the magic basis U is the Kronecker product of two 1-qubit gates, i.e.

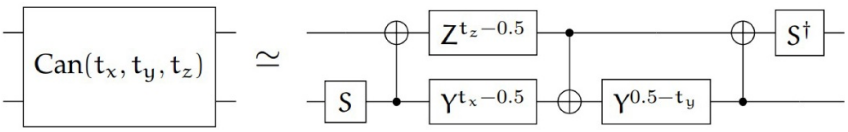
$$V = M\,U\,M^\dagger = A \otimes B \qquad \text{if } U \in SO(4)$$

We assume that the phase has already been extracted and that U is therefore a special unitary matrix. Thus, we can write, $U = (K_3 \otimes K_4)\,\mathrm{Can}(t_x, t_y, t_z)\,(K_1 \otimes K_2)$, and thus:

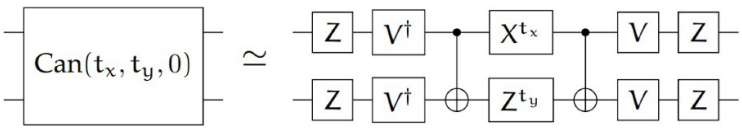
$$V = M\,U\,M^\dagger = M(K_3 \otimes K_4)\,M^\dagger M \mathrm{Can}\left(t_x, t_y, t_z\right) M^\dagger M (K_1 \otimes K_2)\,M^\dagger = Q_2\,D\,Q_1.$$

A transpose of V inverts the orthogonal matrices but leaves the complex diagonal matrix unchanged. Thus, $V^T V = Q_1^T\,D\,Q_2^T Q_2\,D\,Q_1 = Q_1^T\,D\,Q_1$ is a similarity transform of the diagonal matrix squared. An eigen-decomposition of this yields the square eigenvalues of D, and Q_1 as the matrix of eigenvectors. The canonical gate coordinates can then be extracted from the eigenvalues, and the magic basis transform can be undone to recover the local gates. These Kronecker products of local gates can then be decomposed into separate 1-qubit gates using the Kronecker decomposition and further into elementary gates using a 1-qubit decomposition.

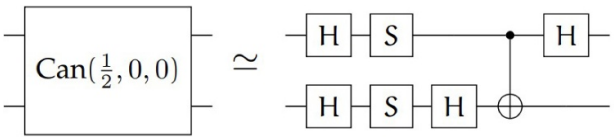
The canonical gate can be further decomposed into CNOT gates or other sets of 2-qubit gates. For example, any general canonical gate can be built from 3 CNOT gates as,



Gates on the bottom surface of the Weyl chamber (special orthogonal local equivalency class) require only 2 CNOT gates,



While gates locally equivalent to CNOT require only one CNOT gate



and those locally equivalent to the identity require none.

A.3. Solovay–Kitaev decomposition

The SKD algorithm [25] applies the Solovay–Kitaev theorem to decompose quantum operators into a sequence of quantum gates drawn from a discrete set. As explained in section 3.1, the algorithm finds approximate sequences such that the length of the decomposed circuit and the runtime of the algorithm scale as a logarithmic factor with the inverse precision $\frac{1}{\epsilon}$. SKD includes a preprocessing step that generates a search space of composite sequences with a maximum length (or depth) d of gates from a finite discrete basis. This set of composite sequences is called the SK-basis. The algorithm functions in a recursive fashion, and the degree of recursion is denoted by n. The algorithm returns a gate sequence that approximates the unitary operator U to an error ϵ_n during the process. The algorithm is designed to obtain an improved approximation accuracy $\epsilon_r < \epsilon_{r-1}$, and eventually, the base case returns the `best_approximation` to a matrix U, bounded by ϵ_0. The approximation accuracy tends to 0 as the recursion depth n increases without bound. In YAQQ, the Qiskit implementation of SKD has been modified to allow arbitrary single-qubit unitary gates (along with their inverse/dagger) to form the gate set

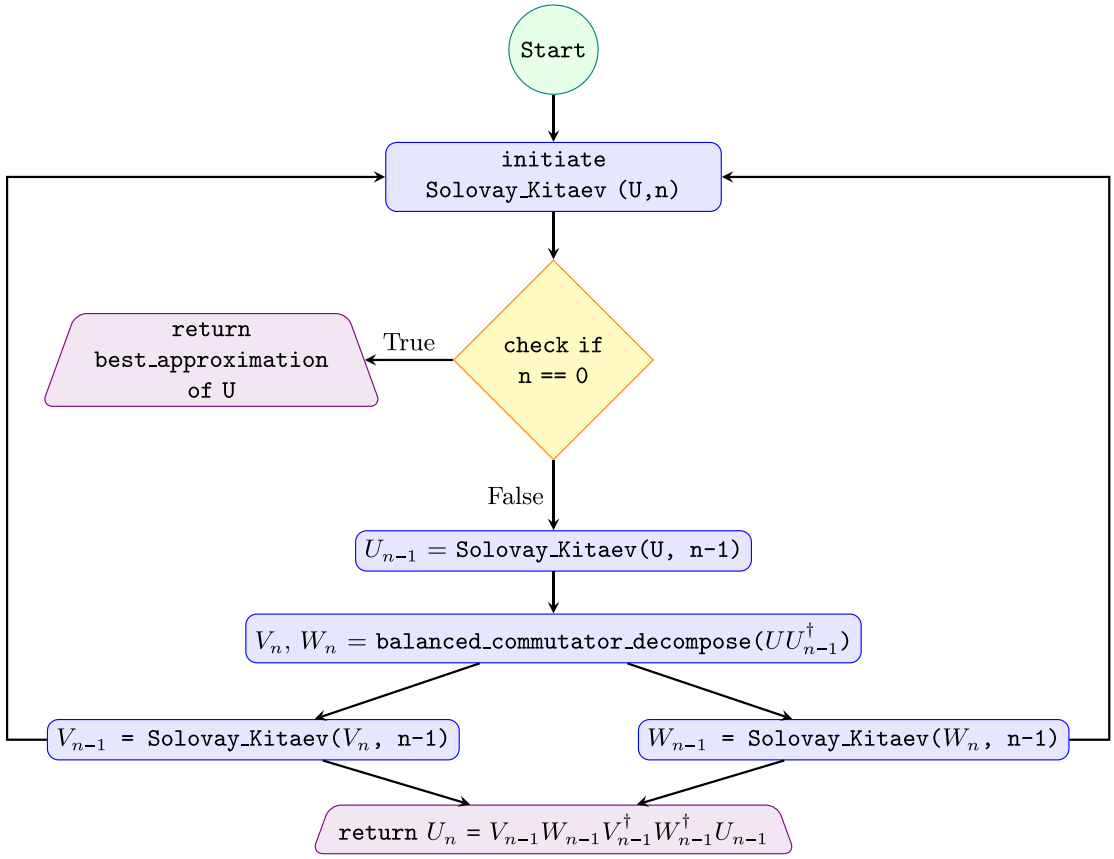


Figure 15. Flowchart of the Solovay–Kitaev Decomposition Algorithm for 1-qubit unitary quantum operator U and recursion depth n. Returns the ϵ_n approximation to the target unitary U computed from call of the function at the $n-1$ degree of recursion, and returns the ϵ_0 approximation in the base case.

for the SK-basis. Such a modification allows us to employ SKD in the DSE of candidate gates in the gate sets.

An extension of the algorithm for single-qubit systems to the general case of m-qubit systems is possible in theory [25] with the modification in only one step of the algorithm—the balanced commutator group decomposition to an approximate version of itself. While this change appears reasonably straightforward, it has a significant non-trivial implementation. The depth of the SK basis also scales exponentially with the system's dimension. Therefore, generalizing SKD to higher dimensions does not function well, necessitating QSD, an exact decomposition technique covered subsequently.

The chosen set of fundamental gates and the group generated by the set, that is, the SK-basis, must fulfill the following conditions generalized for an m-qubit system:

1. All the gates in the set belong to the group of special unitary matrices $SU(m)$ and have a determinant 1.
2. The set of gates is closed under inversion, implying that for every gate in the set, its hermitian conjugate must also belong to the set.
3. The group generated by the set must densely span the space $SU(m)$. This means that, for every arbitrary unitary operation U, there must exist a product sequence of gates from the set that can approximate U with a bounded error ϵ.

The Flowchart 15 explains the implementation of SKD based on the pseudo-code of the algorithm presented in [25] and its Qiskit implementation.

The `balanced_commutator_decompose` method performs a balanced group commutator decomposition of the accuracy at level r defined as $\Delta = UU_{r-1}^{\dagger} = VWV^{\dagger}W^{\dagger}$ for matrices V and W. $(r-1)$ level approximation accuracies are computed by the call of the function again for matrices V and W and the rth level approximate sequence $U_r = V_{r-1}W_{r-1}V_{r-1}^{\dagger}W_{r-1}^{\dagger}U_{r-1}$, consisting of all 5 terms computed form the $(r-1)$th level is returned.

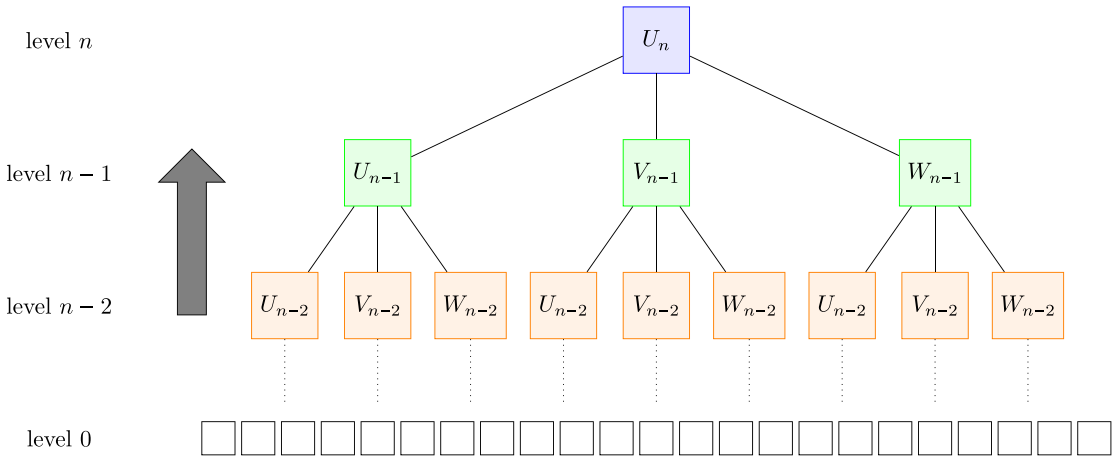


Figure 16. Tree of sequences depicting the working of the Solovay–Kitaev algorithm. Recursion levels are indicated on the left.

An intuitive tree diagram depicting the algorithm's operation is shown in figure 16. The bottom-most layer consists of 3^n blocks and corresponds to the base case of the algorithm. For each block, the `best_approximation` method finds the sequence of gates from the search space that approximates it up to ϵ_0. Each node in the level r, U_r is a composite sequence constructed from its three daughter nodes in the $r-1$ level such that $U_r = V_{r-1}W_{r-1}V_{r-1}^{\dagger}W_{r-1}^{\dagger}U_{r-1}$.

A.4. RD

RD is a flexible alternative for decomposing general unitary matrices using an arbitrary gate set. In RD, a set of random-length sequences is evaluated, with a random sample of gates from the gate set. The sequence with the highest fidelity among the set number of trials is returned as the decomposed circuit. Though it sounds naive, in practice, this performs better than SKT for single-qubit decomposition on most target unitaries. While the advantage is that it can be easily extended to higher-dimensional Hilbert spaces, the number of random trials must be adjusted accordingly and will most likely grow prohibitively for large unitaries.

Appendix B. Definitions of standard built-in gates

The novel gate set ansatz can be specified based on the available gate options, as explained in section 3.2. The exact definitions of these gates are given in table 4.

Table 4. Elements for composing gate sets.

Gate Identifier	Description	Search Method
R1	1-qubit Haar random unitary gate	RND
P1	1-qubit parametric gate (IBM U3)	RND/OPT(3)
T1	1-qubit T gate $:= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$	FXD
TD1	1-qubit T-dagger gate $:= \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/4} \end{bmatrix}$	FXD
S1	1-qubit S gate $:= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{bmatrix}$	FXD
Z1	1-qubit Z gate $:= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	FXD
X1	1-qubit X gate $:= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	FXD
H1	1-qubit Hadamard gate $:= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	FXD
F1	1-qubit unitary gate definition from file	depends
R2	2-qubit Haar random unitary gate	RND
NL2	2-qubit non-local unitary gate	RND/OPT(3)
CX2	2-qubit CNOT gate $:= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	FXD
B2	2-qubit Berkeley gate $:= \begin{bmatrix} \cos(\pi/8) & 0 & 0 & i\sin(\pi/8) \\ 0 & \cos(3\pi/8) & i\sin(3\pi/8) & 0 \\ 0 & i\sin(\pi/8) & \cos(\pi/8) & 0 \\ i\sin(\pi/8) & 0 & 0 & \cos(\pi/8) \end{bmatrix}$	FXD
SPE2	2-qubit special perfect entangler $:= \mathrm{NL2}(0.5, t_y, 0)$	RND/OPT(1)
F2	2-qubit unitary gate definition from file	depends

Appendix C. Software workflow

The overall workflow of YAQQ is shown in figure 17. YAQQ can be run in developer or manual mode. The developer mode needs to specify a configuration file in a specific format that includes all further details, such as the usage mode, data set, gate sets (GS), hyperparameters, and the directory and filename for logging results and plots. In manual mode, the above options can be configured via the command-line interface depending on the usage mode. YAQQ has three usage levels: (i) gate set compiler, (ii) gate set comparator, and (iii) gate set discovery. These three modes depend on one another recursively. In the compiler mode, a gate set and a specific unitary U are given, and YAQQ outputs the decomposed circuit, the associated process fidelity (PF) of the approximation, and the circuit depth (CD). In the comparator mode, two gate sets, {GS1, GS2} and a set of unitaries $\{U\}_n$ is given. The compiler mode is invoked iteratively for each unitary and for both gate sets. The statistics for PF and CD form the basis for the performance comparison. The discovery mode is the core novelty of this research. In this mode, based on a defined cost function, a new gate set, GS2, is iteratively refined for the data set, and another predefined gate set, GS1. This mode, in turn, invokes the comparator iteratively to compare a candidate GS2 against GS1. YAQQ offers various options for visualizing the data set (for 1 and 2 qubits) and for comparing two gate sets for the chosen data set. The plots and the raw data can be stored for reproducibility.

C.1. Software organization and modules

A detailed organization of the module dependencies among the three usage levels is shown in figure 18. The innermost (light green) compiler modules decompose a single unitary into a set of gates. Various decomposition options are available based on the size of the unitary. The decomposition algorithms have associated hyperparameters, such as precision and the number of trials, that the user can change from their default values. The middle comparator level (green) iteratively uses the compiler on a specific dataset and two given gate sets, and compares the statistics for process fidelity and circuit depth. The different dataset options, as discussed previously, are available based on the number of qubits. The user also specifies the associated hyperparameters, like dataset size or resolution. The outer discovery level (darker

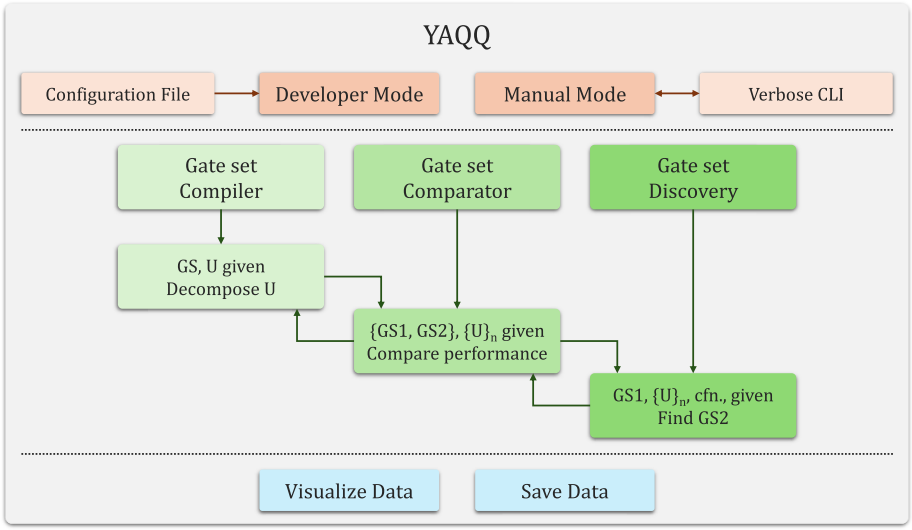


Figure 17. The workflow of YAQQ depicting the three usage levels, (i) gate set compiler, (ii) gate set comparator, and (iii) gate set discovery, and the nested dependencies between these levels.

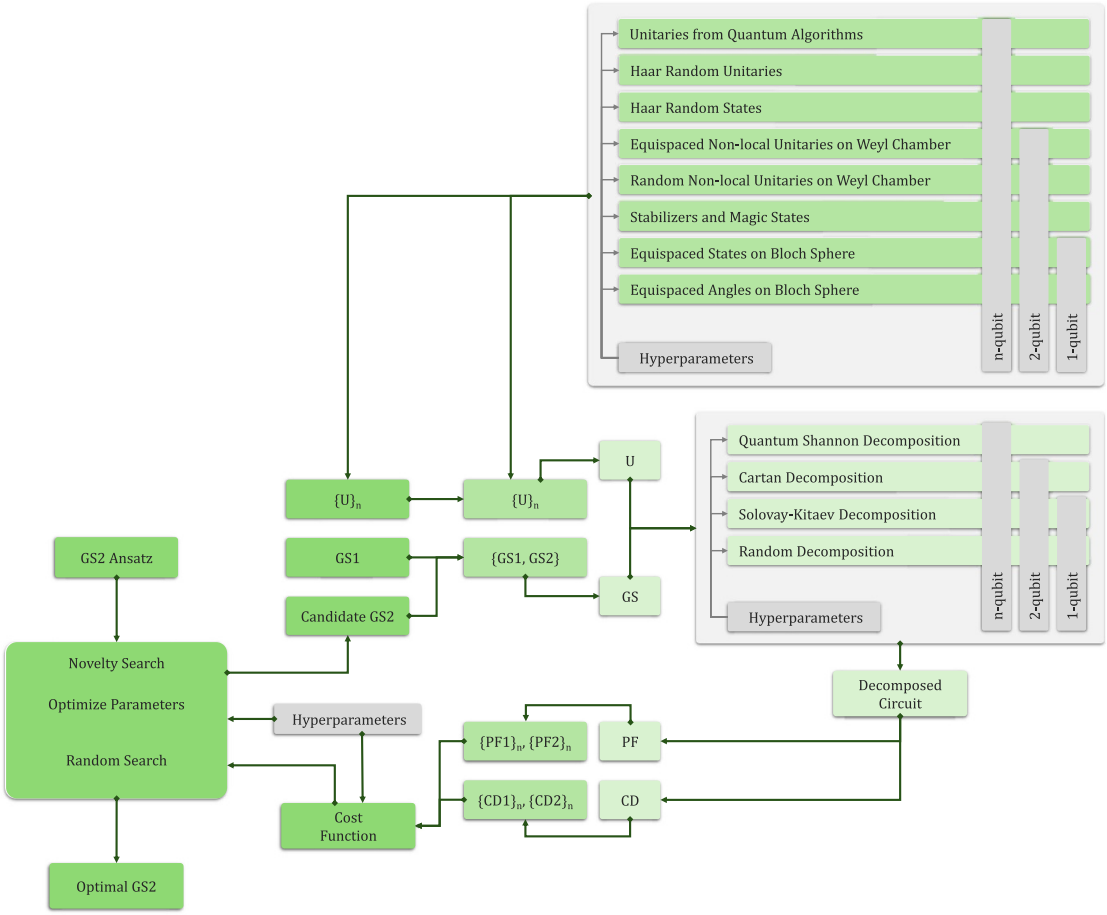


Figure 18. Module dependencies among the three usage levels. The innermost (light green) compiler modules decompose a single unitary into a set of gates. The middle comparator level (green) iteratively uses the compiler for a specific dataset and two given gate sets, and compares the statistics of process fidelity and circuit depth. The outer discovery level (darker green) uses the comparator level iteratively to test a candidate gate set suggested by the novelty search based on the ansatz, and to assess it via a tunable cost function. A new optimal gate set with respect to the given gate set (for the chosen data set and cost function) is discovered in the process.

green) uses the comparator level iteratively to test a candidate gate set suggested by the novelty search based on the ansatz, and to assess it via a tunable cost function. The hyperparameters at this level define the weights in the cost function, the optimization algorithm to use, and associated parameters. A novel optimal gate set, conditioned on the given gate set, the chosen data set, and the cost function, is discovered in the process.

ORCID iDs

Akash Kundu 0000-0002-3540-1061
Sibasish Mishra 0000-0002-3756-8420
Jarosław A Miszczak 0000-0001-8790-101X
Sebastian Feld 0000-0003-2782-1469

References

[1] Preskill J 2018 Quantum computing in the NISQ era and beyond *Quantum* **2** 79
[2] Bharti K *et al* 2022 Noisy intermediate-scale quantum algorithms *Rev. Mod. Phys.* **94** 015004
[3] Bernstein E and Vazirani U 1993 Quantum complexity theory *Proc. 25th Annual ACM Symp. on Theory of Computing* pp 11–20
[4] Shor P W 1999 Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer *SIAM Rev.* **41** 303–32
[5] Grover L K 1996 A fast quantum mechanical algorithm for database search *Proc. 28th Annual ACM Symp. on Theory of Computing* pp 212–9
[6] De Wetering J V 2020 ZX-calculus for the working quantum computer scientist (arXiv:2012.13966)
[7] Sharma K, Khatri S, Cerezo M and Coles P J 2020 Noise resilience of variational quantum compiling *New J. Phys.* **22** 043006
[8] Chi-Kwong Li, Roberts R and Yin X 2013 Decomposition of unitary matrices and quantum gates *Int. J. Quantum Inf.* **11** 1350015
[9] Sarra L, Ellis K and Marquardt F 2024 Discovering quantum circuit components with program synthesis *Mach. Learn.: Sci. Technol.* **5** 025029
[10] Cross A W, Bishop L S, Sheldon S, Nation P D and Gambetta J M 2019 Validating quantum computers using randomized model circuits *Phys. Rev. A* **100** 032328
[11] Donkers H, Mesman K, Al-Ars Z and Möller M 2022 Qpack scores: quantitative performance metrics for application-oriented quantum computer benchmarking (arXiv:2205.12142)
[12] Martiel S, Ayral T and Allouche C 2021 Benchmarking quantum coprocessors in an application-centric, hardware-agnostic and scalable way *IEEE Trans. Quantum Eng.* **2** 1–11
[13] Mykhailova M and Soeken M 2021 Testing quantum programs using q# and microsoft quantum development kit *Proc. of the 2nd Quantum Software and Engineering Workshop (QSET'21)* pp 81–88 (available at: https://ceur-ws.org/Vol-3008/short6.pdf)
[14] Quetschlich N, Burgholzer L and Wille R 2023 MQT Bench: benchmarking software and design automation tools for quantum computing *Quantum* **7** 1062
[15] Barenco A, Bennett C H, Cleve R, DiVincenzo D P, Margolus N, Shor P, Sleator T, Smolin J A and Weinfurter H 1995 Elementary gates for quantum computation *Phys. Rev. A* **52** 3457
[16] Parzanchevski O and Sarnak P 2018 Super-golden-gates for pU (2) *Adv. Math.* **327** 869–901
[17] Kuperberg G 2023 Breaking the cubic barrier in the Solovay-Kitaev algorithm (arXiv:2306.13158)
[18] Lin S F, Sussman S, Duckering C, Mundada P S, Baker J M, Kumar R S, Houck A A and Chong F T 2022 Let each quantum bit choose its basis gates *2022 55th IEEE/ACM Int. Symp. on Microarchitecture (MICRO)* (IEEE) pp 1042–58
[19] Davis M, Iancu C, Smith E and Younis E 2022 Gate-set exploration using high-quality numerical synthesis *APS March Meeting Abstr.* vol **2022** pp G37–002 (available at: https://ui.adsabs.harvard.edu/abs/2022APS..MARG37002D)
[20] Kalloor J, Weiden M, Younis E, Kubiatowicz J, De Jong B and Iancu C 2024 Quantum hardware roofline: evaluating the impact of gate expressivity on quantum processor design *2024 Int. Conf. on Quantum Computing and Engineering* (https://doi.org/10.1109/QCE60285.2024.00100)
[21] Johnson S C *et al* 1975 *Yacc: Yet Another Compiler-Compiler* vol 32 (Bell Laboratories Murray Hill, NJ) (available at: https://epaperpress.com/lexandyacc/download/yacc.pdf)
[22] Khatri S, LaRose R, Poremba A, Cincio L, Sornborger A T and Coles P J 2019 Quantum-assisted quantum compiling *Quantum* **3** 140
[23] Jones T and Benjamin S C 2022 Robust quantum compilation and circuit optimisation via energy minimisation *Quantum* **6** 628
[24] Lehman J and Stanley K O 2011 Abandoning objectives: evolution through the search for novelty alone *Evolutionary comput.* **19** 189–223
[25] Dawson C M and Nielsen M A 2005 The Solovay-Kitaev algorithm (arXiv:quant-ph/0505030)
[26] Qiskit contributors 2023 Qiskit: an open-source framework for quantum computing (https://doi.org/10.5281/zenodo.2562111)
[27] Zhang S-X, Hsieh C-Y, Zhang S and Yao H 2022 Differentiable quantum architecture search *Quantum Sci. Technol.* **7** 045023
[28] Hai V T, Viet N T, Urbaneja J, Linh N V, Tran L N and Le Bin H 2024 Multi-target quantum compilation algorithm *Mach. Learn.: Sci. Technol.* **5** 045057
[29] Rattacaso D, Jaschke D, Ballarin M, Siloi I and Montangero S 2025 Quantum circuit compilation with quantum computers *Phys. Rev. Res.* **7** 033268
[30] Kundu A 2025 awesome-QAS: a curated list of resources for quantum architecture search (available at: https://github.com/Aqasch/awesome-QAS) (Accessed 05 March 2026)
[31] Kundu A and Sarra L 2026 Reinforcement learning with learned gadgets to tackle hard quantum problems on real hardware *Commun. Phys.* **9** 44
[32] Ostaszewski M, Trenkwalder L M, Masarczyk W, Scerri E and Dunjko V 2021 Reinforcement learning for optimization of variational quantum circuit architectures *Advances in Neural Information Processing Systems* vol 34 pp 18182–94

[33] Fösel T, Niu M Y, Marquardt F and Li Li 2021 Quantum circuit optimization with deep reinforcement learning (arXiv:2103.07585)
[34] Patel Y J, Kundu A, Ostaszewski M, Bonet-Monroig X, Dunjko V and Danaci O 2024 Curriculum reinforcement learning for quantum architecture search under hardware errors *The 12th Int. Conf. on Learning Representations* (available at: https://openreview.net/pdf?id=rINBD8jPoP)
[35] Younis E, Iancu C C, Lavrijsen W, Davis M and Smith E 2021 Berkeley quantum synthesis toolkit (bqskit) v1 *Technical Report* (Lawrence Berkeley National Laboratory (LBNL)) (https://doi.org/10.11578/dc.20210603.2)
[36] Younis E, Sen K, Yelick K and Iancu C 2020 Qfast: quantum synthesis using a hierarchical continuous circuit space (arXiv:2003.04462)
[37] Campbell C *et al* 2023 Superstaq: deep optimization of quantum programs *2023 IEEE Int. Conf. on Quantum Computing and Engineering (QCE)* vol 1 (IEEE) pp 1020–32
[38] Gold E M 1967 Language identification in the limit *Inf. Control* **10** 447–74
[39] Lehman J and Stanley K O 2010 Efficiently evolving programs through the search for novelty *Proc. 12th Annual Conference on Genetic and Evolutionary Computation* pp 837–44
[40] Baldassarre G *et al* 2013 *Intrinsically Motivated Learning in Natural and Artificial Systems* (Springer) (https://doi.org/10.1007/978-3-642-32375-1)
[41] Mouret J-B 2011 Novelty-based multiobjectivization *New Horizons in Evolutionary Robotics: Extended Contributions From the 2009 Evoderob Workshop* (Springer) pp 139–54
[42] Fauquenot S, Sarkar A and Feld S 2025 Open and closed loop approaches for energy efficient quantum optimal control *Adv. Quantum Technol.* **8** 2400690
[43] Steinberg M, Bandić M, Szkudlarek S, Almudever C G, Sarkar A and Feld S 2024 Lightcone bounds for quantum circuit mapping via uncomplexity *npj Quantum Inf.* **10** 113
[44] Lloyd S 1995 Almost any quantum logic gate is universal *Phys. Rev. Lett.* **75** 346
[45] Krol A M and Al-Ars Z 2024 Highly efficient decomposition of n-qubit quantum gates based on block-ZXZ decomposition (arXiv:2403.13692)
[46] Shende V V, Markov I L and Bullock S S 2004 Minimal universal two-qubit controlled-not-based circuits *Phys. Rev. A* **69** 062321
[47] Shannon C E 1949 The synthesis of two-terminal switching circuits *Bell Syst. Tech. J.* **28** 59–98
[48] Shende V V, Bullock S S and Markov I L 2005 Synthesis of quantum logic circuits *Proc. 2005 Asia and South Pacific Design Automation Conf.* pp 272–5
[49] King Yiu Y, Sarkar A, Rimbach-Russ M, Ishihara R and Feld S 2025 Transformer models for quantum gate set tomography *Quantum Mach. Intell.* **7** 10
[50] Eastin B and Knill E 2009 Restrictions on transversal encoded quantum gate sets *Phys. Rev. Lett.* **102** 110502
[51] Sayginel H, Jamet F, Agarwal A, Browne D E and Rungger I 2023 A fault-tolerant variational quantum algorithm with limited T-depth *Quantum Sci. Technol.* **9** 015015
[52] Wolfram S 2002 *A New Kind of Science* (Wolfram Media) (available at: https://store.wolfram.com/view/book/ISBN1579550088.str?Qualifier=COMM)
[53] Ozols M 2009 The solovay-litaev theorem *Essay at* (University of Waterloo) (available at: http://home.lu.lv/~sd20008/papers/essays/Solovay-Kitaev.pdf)
[54] Trung P T, Van Meter R and Horsman C 2012 Optimising the Solovay-Kitaev algorithm *Phys. Rev. A* **87** 052332
[55] DiVincenzo D P 1995 Two-bit gates are universal for quantum computation *Phys. Rev. A* **51** 1015
[56] Crooks G E 2020 Gates, states and circuits *Gates states circuits* pp 1–80 (available at: https://chunyangding.com/assets/files/on-gates.pdf)
[57] Trenkwalder L M, López Incera A, Nautrup H P, Flamini F and Briegel H J 2022 Automated gadget discovery in science (arXiv:2212.12743)
[58] Sarra L, Ellis K and Marquardt F 2023 Discovering quantum circuit components with program synthesis *Mach. Learn.: Sci. Technol.* **5** 025029
[59] Sarkar A, Al-Ars Z and Bertels K 2022 Qksa: quantum knowledge seeking agent *Int. Conf. on Artificial General Intelligence* (Springer) pp 384–93
[60] Vitányi P M B 2001 Quantum Kolmogorov complexity based on classical descriptions *IEEE Trans. Inf. Theory* **47** 2464–79
[61] Berthiaume A, Van Dam W and Laplante S 2001 Quantum Kolmogorov complexity *J. Comput. Syst. Sci.* **63** 201–21
[62] Mora C E, Briegel H J and Kraus B 2007 Quantum Kolmogorov complexity and its applications *Int. J. Quantum Inf.* **5** 729–50
[63] Kaltchenko A 2021 Kolmogorov complexity of unitary transformations in quantum computing (arXiv:2110.05937)
[64] Sarkar A, Al-Ars Z, and Bertels K 2020 Quantum circuit design for universal distribution using a superposition of classical automata (arXiv:2006.00987)
[65] Xie S, Sarkar A, and Feld S 2025 Deqompile: quantum circuit decompilation using genetic programming for explainable quantum architecture search (arXiv:2504.08310)
[66] Bach B G, Kundu A, Acharya T and Sarkar A 2023 Visualizing quantum circuit probability: estimating quantum state complexity for quantum program synthesis *Entropy* **25** 763
[67] Schaffer C 1994 A conservation law for generalization performance *Machine Learning Proc. 1994* (Elsevier) pp 259–65
[68] Wolpert D H and Macready W G 1997 No free lunch theorems for optimization *IEEE Trans. Evolutionary Comput.* **1** 67–82
[69] Maziero J 2015 Random sampling of quantum states: a survey of methods: and some issues regarding the overparametrized method *Braz. J. Phys.* **45** 575–83
[70] IBM. qiskit.circuit.library U3Gate. 2013 (available at: https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.U3Gate) (Accessed 05 February 2024)
[71] Powell M J D 1994 A direct search optimization method that models the objective and constraint functions by linear interpolation *Advances in Optimization and Numerical Analysis* (Springer) pp 51–67
[72] Bonet-Monroig X, Wang H, Vermetten D, Senjean B, Moussa C, Bäck T, Dunjko V and O'Brien T E 2023 Performance comparison of optimization methods on variational quantum algorithms *Phys. Rev. A* **107** 032407
[73] Yaqq pypi (available at: https://pypi.org/project/yaqq/) (Accessed 01 January 2024)
[74] Bergstra J and Bengio Y 2012 Random search for hyper-parameter optimization *J. Mach. Learn. Res.* **13** 281–305

[75] Feurer M and Hutter F 2019 Hyperparameter optimization *Automated Machine Learning: Methods, Systems, Challenges* (Springer) pp 3–33
[76] Wong J, Manderson T, Abrahamowicz M, Buckeridge D L and Tamblyn R 2019 Can hyperparameter tuning improve the performance of a super learner?: a case study *Epidemiology* **30** 521–31
[77] Pearson K 1895 Vii. note on regression and inheritance in the case of two parents *Proc. R. Soc. London* **58** 240–2
[78] Lidar D A and Brun T A 2013 *Quantum Error Correction* (Cambridge University Press) (available at: https://www.cambridge.org/nl/universitypress/subjects/physics/quantum-physics-quantum-information-and-quantum-computation/quantum-error-correction?format=HB&isbn=9780521897877)
[79] Howard M and Campbell E 2017 Application of a resource theory for magic states to fault-tolerant quantum computing *Phys. Rev. Lett.* **118** 090501
[80] Anderson J T, Duclos-Cianci G and Poulin D 2014 Fault-tolerant conversion between the steane and reed-muller quantum codes *Phys. Rev. Lett.* **113** 080501
[81] Xiang F *et al* 2019 eQASM: An executable quantum instruction set architecture *2019 IEEE Int. Symp. on High Performance Computer Architecture (HPCA)* (IEEE) pp 224–37
[82] Słowik O and Sawicki A 2023 Calculable lower bounds on the efficiency of universal sets of quantum gates *J. Phys. A: Math. Theor.* **56** 115304
[83] Blackman T R and Stier Z 2022 Fast navigation with icosahedral golden gates (arXiv:2205.03007)
[84] Rezakhani A T 2004 Characterization of two-qubit perfect entanglers *Phys. Rev. A* **70** 052313
[85] Dalal R, Evra S, and Parzanchevski O 2025 Multi-qubit golden gates (arXiv:2509.09047)
[86] Cruz-Lemus J A, Marcelo L A and Piattini M 2021 Towards a set of metrics for quantum circuits understandability *Int. Conf. on the Quality of Information and Communications Technology* (Springer) pp 239–49
[87] Wyeth C and Sturtivant C 2023 A circuit complexity formulation of algorithmic information theory *Physica D* **456** 133925
[88] Abrahão F S and Zenil H 2022 Emergence and algorithmic information dynamics of systems and observers *Phil. Trans. R. Soc. A* **380** 20200429
[89] Svozil K 1995 Quantum algorithmic information theory (arXiv:quant-ph/9510005)
[90] Mishra S, Sarkar A and Feld S 2026 EQISA: energy-efficient quantum instruction set architecture using sparse dictionary learning (arXiv:2603.20646)
[91] Yuxuan D, Huang T, You S, Hsieh M-H and Tao D 2022 Quantum circuit architecture search for variational quantum algorithms *npj Quantum Inf.* **8** 62
[92] Kundu A, Bedelek P, Ostaszewski M, Danaci O, Patel Y J, Dunjko V and Miszczak J 2024 Enhancing variational quantum state diagonalization using reinforcement learning techniques *New J. Phys.* **26** 013034
[93] Mitarai K, Negoro M, Kitagawa M and Fujii K 2018 Quantum circuit learning *Phys. Rev. A* **98** 032309
[94] Sadhu A, Sarkar A and Kundu A 2024 A quantum information theoretic analysis of reinforcement learning-assisted quantum architecture search *Quantum Mach. Intell.* **6** 49
[95] Kundu A, Sarkar A and Sadhu A 2024 Kanqas: Kolmogorov-arnold network for quantum architecture search *EPJ Quantum Technol.* **11** 76
[96] Sadhu A, Sarkar A, and Kundu A 2025 Cutqas: topology-aware quantum circuit cutting via reinforcement learning (arXiv:2504.04167)
[97] Apak B, Bandic M, Sarkar A and Feld S 2024 Ketgpt–dataset augmentation of quantum circuits using transformers *Int. Conf. on Computational Science* (Springer) pp 235–51
[98] Azses D, Dupont M, Evert B, Reagor M J and Torre E G D 2023 Navigating the noise-depth tradeoff in adiabatic quantum circuits *Phys. Rev. B* **107** 125127
[99] Qian W, Xin Li, Riedel M D, Bazargan K and Lilja D J 2010 An architecture for fault-tolerant computation with stochastic logic *IEEE Trans. Comput.* **60** 93–105
[100] Sarkar A and Kundu A 2023 An agent that searches for novel quantum universal gate set *Yet Another Quantum Quantizer* (available at: https://github.com/Advanced-Research-Centre/YAQQ)
[101] Krol A M, Sarkar A, Ashraf I, Al-Ars Z and Bertels K 2022 Efficient decomposition of unitary matrices in quantum circuit compilers *Appl. Sci.* **12** 759
[102] Tucci R R 1999 A rudimentary quantum compiler (2cnd ed.) (arXiv:quant-ph/9902062)
[103] Tucci R R 2005 An introduction to Cartan's KAK decomposition for QC programmers (arXiv:quant-ph/0507171)
[104] Khaneja N and Glaser S J 2001 Cartan decomposition of SU (2n) and control of spin systems *Chem. Phys.* **267** 11–23
[105] Kraus B and Cirac J I 2001 Optimal creation of entanglement using a two-qubit gate *Phys. Rev. A* **63** 062309