IOP Publishing Quantum Sci. Technol. 10 (2025) 025029 https://doi.org/10.1088/2058-9565/adb3c7

Quantum Science and Technology

OPEN ACCESS

RECEIVED
8 November 2024

REVISED
20 January 2025

ACCEPTED FOR PUBLICATION
7 February 2025

PUBLISHED
18 February 2025

PAPER

Encoding proteins as quantum states with approximate quantum state preparation by iterated sparse state preparation

Rod Rofougaran[1], **Ralph Wang**[1], **Akshay Ajagekar**[1] **and Fengqi You**[2,3,*]

[1] School of Applied and Engineering Physics, Cornell University, Ithaca, NY, United States of America
[2] Systems Engineering, Cornell University, Ithaca, NY, United States of America
[3] Robert Frederick Smith School of Chemical and Biomolecular Engineering, Cornell University, Ithaca, NY, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: fengqi.you@cornell.edu

Keywords: quantum state preparation, algorithm, protein

Supplementary material for this article is available online

Abstract

Quantum computing holds transformative potential for various domains including cheminformatics through advancements in quantum algorithms. The key to realizing improvements with quantum algorithms in cheminformatics is encoding chemical data like proteins as quantum states with quantum state preparation. In this work, we propose a computational framework to encode proteins as quantum states for efficient downstream quantum processing. Protein data representations are encoded as multi-qubit quantum states with iterative quantum sparse state preparation guided by the classical heuristic search method for optimal gate sequence identification. The validity and efficiency of the proposed method is demonstrated with various computational experiments to encode uniform random states as well as proteins. Several performance comparisons against the baselines of exact and variational state preparation methods, the proposed approach is able to encode proteins with 25% fewer controlled-NOT gates while performing orders of magnitude faster than the variational method.

1. Introduction

Quantum algorithms show potential for speeding up computational routines, such as the quantum Fourier transform [1], simulating quantum systems [2–6], and solving linear systems [7, 8]. In addition, quantum machine learning (QML) has demonstrated potential to advance protein design and structure prediction, which can mitigate crises in resource management, healthcare, and sustainability [9–12]. Encoding chemical data onto quantum computers is a crucial first step in leveraging their potential because it enables the quantum system to directly manipulate and analyze data reflecting the underlying properties of the molecules themselves [13]. The quantum state preparation (QSP) subroutine, which prepares a multi-qubit quantum state that encodes classical data, is an essential but often expensive step in such quantum algorithms and machine learning applications [14]. Accurate QSP is essential for developing quantum algorithms that can exploit any advantages [15] to solve problems like learning from chemical data like proteins to realize goals like capturing structure-property relationships.

Several exact methods for QSP have been previously proposed, such as the uniformly controlled gates (UCGs) approach [16] and its various improvements [17–20]. Many sparse state preparation methods have also been investigated [21, 22], however, they generally assume that CX gates can be applied to any qubit pair. An alternative to exact QSP methods is approximate encoding using variational quantum circuits (VQCs) [23, 24], which is hardware-adaptive and requires fewer CX gates compared to the exact QSP approach. Extracting optimal gate parameters for VQCs can be plagued by the barren plateau problem caused by vanishing gradients [23, 25]. While exact QSP methods tend to use larger gate counts compared to VQC-based methods, VQC-based methods require more computational resources to generate a gate

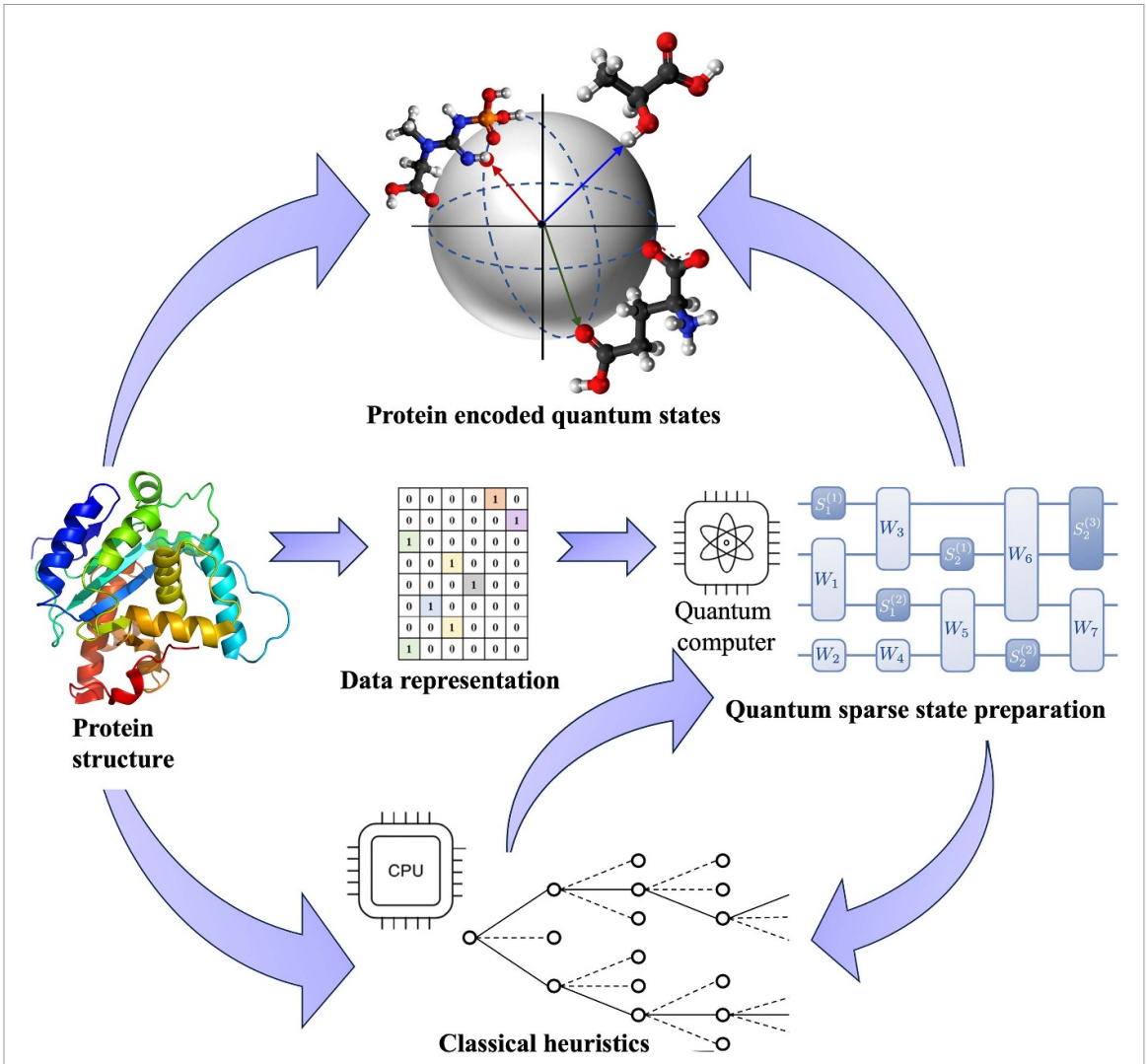

Figure 1. An overview of the framework for encoding proteins as quantum states based on protein data representations, quantum sparse state preparation, and classical heuristics for refinement.

sequence. Additionally, QSP using matrix product states (MPSs) has been shown to efficiently prepare quantum states which encode smooth functions, such as probability distributions [26–28].

Efficient preparation of quantum states encoded with chemical data is crucial for applications like QML in cheminformatics that impact both the performance and the feasibility of applications such as property prediction tasks [11, 29]. Furthermore, efficient state preparation reduces the computational overhead associated with initiating QML tasks [30]. As near-term quantum computers suffer from short coherence times and noisy quantum gate operations [31], executing quantum algorithms with near-term hardware requires an efficient QSP implementation [14]. This efficiency is particularly crucial in cheminformatics [32], so effective state preparation can thereby enhances the feasibility of applying quantum computing to real-world problems involving large-scale protein data.

Realizing the framework for efficient QSP implementation of protein data requires addressing a few research challenges revolving around mitigating the cost of a quantum circuit. The first research challenge lies in minimizing the quantum circuit cost measured using gate count (two-qubit gates) or circuit depth [16], as larger circuit depth can lead to qubit decoherence [19]. As many quantum architectures follow a linear nearest neighbor (LNN) architecture wherein the ith qubit's nearest neighbors are the $i-1$th and $i+1$th qubit [33, 34], ensuring protein-encoded QSP for LNN by taking hardware connectivity into account is another research challenge. Another research challenge lies in identifying a heuristic to find short gate sequences that approximately prepare quantum states.

To address these research challenges, we propose a framework for efficiently preparing quantum states encoded with chemical data like proteins, as shown in figure 1. The proposed approach consists of three primary components: protein embeddings generated by pre-trained language models, quantum sparse state preparation circuits, and classical heuristic search methods, which work in synergy to streamline the

preparation process. Protein embeddings capture information about amino acid sequences and molecular properties and form the raw data encoded onto quantum states via amplitude encoding. Our framework can be extended to prepare states encoding other types of biomolecular data, such as DNA or metabolic structures, with variations only in the embedding dimensions and the number of qubits required to represent the state. State-of-the-art machine learning models typically generate low dimensional biomolecular embeddings with dimensions ranging from a few dozens to a few thousand [35–40], which can be efficiently encoded into quantum states of 14 qubits or fewer. Another core component of the proposed framework is the application of quantum circuits capable of preparing sparse quantum states. Sparse quantum states on n qubits have at most m nonzero amplitudes, where $m \ll 2^n$ [41]. The efficient preparation of sparse quantum states finds applications in solving linear systems [7], the quantum Byzantine agreement [42], and other quantum algorithms [43, 44]. The third key element is a classical heuristic search method for identifying optimal sparse QSP circuits, defined by their gate sequences and parameters, which can be iteratively applied to prepare general quantum states.

To demonstrate the applicability of our method to prepare quantum states in both general and machine learning applications, we validate our method with randomly sampled quantum states up to 14 qubits and ten-qubit quantum states that encode proteins from the UniProtKB database [45]. We compared the CX gate count and classical runtime of our iterated sparse approximation (ISA) implementation against that of UCG and two VQC-based methods. In addition, we studied the relationship between theoretical and observed preparation fidelity by performing noisy simulations of our algorithm on five-qubit states. The major contributions of this work are:

- A novel, hardware-adaptive, non-variational framework for approximate QSP is proposed.
- A connection between sparse QSP and general QSP is established.
- A specific implementation of the proposed framework is described and shown to effectively prepare both generic quantum states and those representing real-world data.

2. ISA framework

2.1. QSP

QSP is formally defined as: given an arbitrary quantum state $|x\rangle$ and a family of quantum gates G, return a sequence of quantum gates $h_1, h_2, \ldots, h_m$ from G such that $h_m \ldots h_2 h_1 |\mathbf{0}\rangle = |x\rangle$ [16]. In this paper, we assume G contains continuously parameterized RX, RY and RZ gates, as well as the CX gate. This gate set is chosen because these gates are both easy to reason about and easy to compile into the native gate set of existing quantum computers [46]. In this work, we tackle the following version of approximate QSP: given an arbitrary quantum state $|x\rangle$, return a sequence of quantum gates $g_1, g_2, \ldots, g_m$ that satisfies equation (1),

$$g_m \ldots g_2 g_1 |x\rangle = |y\rangle \tag{1}$$

$$|\langle y|\mathbf{0}\rangle|^2 \geqslant 1 - \epsilon \tag{2}$$

$$\begin{aligned} |\langle x|x'\rangle|^2 &= |\langle x|g_1^\dagger g_2^\dagger \ldots g_m^\dagger|\mathbf{0}\rangle|^2 \\ &= |\langle y|\mathbf{0}\rangle|^2 \geqslant 1 - \epsilon. \end{aligned} \tag{3}$$

Equation (2) represents the state preparation fidelity requirement with ϵ as the fidelity error.

This version of approximate QSP can be applied towards approximately preparing arbitrary quantum states. For some state $|x\rangle$, if a sequence of gates can be found for transforming $|x\rangle$ to $|y\rangle$, such that $|y\rangle$ is close to $|\mathbf{0}\rangle$, then inverting each gate and reversing the gate sequence generates a gate sequence for approximately preparing $|x\rangle$ starting from $|\mathbf{0}\rangle$. Indeed, if $|y\rangle = g_m \ldots g_2 g_1 |x\rangle$, $|\langle \mathbf{0}|y\rangle|^2 \geqslant 1 - \epsilon$, and $|x'\rangle = g_1^\dagger g_2^\dagger \ldots g_m^\dagger |\mathbf{0}\rangle$, then equation (3) follows. This indicates that the prepared state $|x'\rangle$ is a good approximation of the target state $|x\rangle$. For this work, qubit indices start at zero and start from the right. This means $|0010\rangle$ is the result of applying an X gate to qubit 1 of $|0000\rangle$. Single-qubit rotations are written with the rotation angle first, then the qubit index. For example, an RZ gate, angle $\frac{\pi}{2}$, applied to qubit 1, is written as $RZ(\frac{\pi}{2}, 1)$. CX gates are written with the control qubit first. For example, a CX gate applied to qubit 2, with qubit 1 as the control, is written as $CX(1, 2)$. Unless otherwise specified, n will represent the number of qubits in the system and $N = 2^n$ will represent the number of amplitudes to be encoded onto those n qubits. In addition, $|0\rangle^{\otimes n}$, the starting state of an n-qubit quantum processor, will be abbreviated as $|\mathbf{0}\rangle$.

2.2. Patterns and substates

A k-bit pattern denoted by p over n-qubits is defined as a length-n string $\{0,1,*\}^n$ containing exactly k '$*$' characters. The parameter k may be zero, in which case p would be a length-n bit-string. Let I_p denote the set of 2^k integers such that, integer $i \in I_p$ is a permutation of the k-bit pattern formed by replacing $*$ with 0 or 1. Next, define the p-substate of a quantum state $|x\rangle$ with the operator ξ_p as shown in equation (4).

$$\xi_p|x\rangle = \sum_{i \in I_p} |i\rangle\langle i|x\rangle. \tag{4}$$

When $|x\rangle$ is written in the computational basis, $\xi_p|x\rangle$ contains the subset of basis states $|i\rangle$ where i matches p. In most cases, the norm of $\xi_p|x\rangle$ is less than 1, and therefore does not represent a proper quantum state, but is instead a complex-valued vector. Still, we define the action of quantum gates on these substates as we would for normalized quantum states. Substates are defined analogously for quantum state-derived functionals as $(\xi_p|x\rangle)^\dagger = \langle x|\xi_p$.

An X or CX gate may be applied to a pattern p as long as their relevant qubit indices do not correspond to indices where p has a '$*$'. If an X gate targeting qubit i is applied to a pattern p, the result is p but with the ith character inverted from 0 to 1 or vice versa. If a CX gate with control qubit i and target qubit j is applied to p and $p_i = 0$, the result is p; otherwise, the result is p but with the jth character inverted. This behavior of X and CX gates on patterns is chosen to match the behavior of those gates on substates, as described in lemma A1 in appendix A; because of this relationship, we let $g(p)$ denote the result of applying a gate g to a pattern p.

2.3. Substate merging

Our implementation of the ISA framework makes extensive use of a substate merging subroutine, which we describe here. Given a quantum state $|x\rangle$, a k-bit pattern p, and an X gate g that can be applied to p that follows $g(p) \neq p$, the substate merging subroutine applies a single qubit rotation R to $|x\rangle$ such that the norm of $\xi_p R|x\rangle$ is maximized. Any single qubit rotation can be decomposed into an RZ–RY–RZ sequence. Since RZ gates only apply complex phases to the amplitudes in the computational basis, the last RZ has no effect on the norm of $\xi_p R|x\rangle$ and can be left out. Thus, we parameterize $R = RY(\theta)RZ(\phi)$. To compute the optimal θ and ϕ, we perform casework on $p[i]$. If $p[i] = 0$, then $g(p)[i] = 1$, while the norm objective function can be denoted by:

$$\begin{aligned}\xi_p R|x\rangle &= \xi_p RY(\theta)\, RZ(\phi)\, |x\rangle \\ &= \cos\left(\frac{\theta}{2}\right)\xi_p|x\rangle - e^{i\phi}\sin\left(\frac{\theta}{2}\right)\xi_p g|x\rangle \quad \text{(ignoring global phase)}.\end{aligned} \tag{5}$$

The squared norm of this vector is then:

$$\begin{aligned}|\xi_p R|x\rangle|^2 &= \cos^2\left(\frac{\theta}{2}\right)\langle x|\xi_p|x\rangle - 2\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\theta}{2}\right)\mathrm{Re}\left(e^{-i\phi}\langle x|g^\dagger\xi_p|x\rangle\right. + \sin^2\left(\frac{\theta}{2}\right)\langle x|_p g^\dagger\xi_p g|x\rangle \\ &= \cos^2\left(\frac{\theta}{2}\right)\langle x|\xi_p|x\rangle - 2\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\theta}{2}\right)\mathrm{Re}\left(e^{-i\phi}\langle x|\xi_{g(p)}g\xi_p|x\rangle\right) + \sin^2\left(\frac{\theta}{2}\right)\langle x|\xi_{g(p)}|x\rangle\end{aligned} \tag{6}$$

where the last step used lemma A1 to simplify. The first and third terms of this expression are non-negative, and do not depend on the sign of θ. Therefore, θ can be chosen such that the coefficient $2\cos(\frac{\theta}{2})\sin(\frac{\theta}{2})$ of the second term is negative. Following this, the expression is maximized when $e^{-i\phi}\langle x|\xi_{g(p)}g\xi_p|x\rangle$ is a positive real number with no imaginary part, which is achieved by assigning $\phi = \mathrm{phase}(\langle x|\xi_{g(p)}g\xi_p|x\rangle)$. Performing this substitution and simplifying using half-angle laws yields equation (7). This quantity is maximized when θ satisfies equation (8),

$$|(R|x\rangle)_p|^2 = \frac{\langle x|\xi_p|x\rangle + \langle x|\xi_{g(p)}|x\rangle}{2} + \frac{\cos(\theta)}{2}\left\{\langle x|\xi_p|x\rangle - \langle x|\xi_{g(p)}|x\rangle\right\} - \sin(\theta)\,|\langle x|\xi_{g(p)}\xi_p|x\rangle| \tag{7}$$

$$\theta = \tan^{-1}\frac{-2|\langle x|\xi_{g(p)}\xi_p|x\rangle|}{\langle x|\xi_p|x\rangle - \langle x|\xi_{g(p)}|x\rangle}. \tag{8}$$

Alternatively, in the case of $p[i] = 1$ and $g(p)[i] = 0$, $\xi_p R|x\rangle$ follows equation (9). ignoring global phase. Performing the same calculations as above, the magnitude of this state is maximized when conditions in equation (10) are satisfied. By determining θ and ϕ, R can be constructed. This procedure of determining θ and ϕ to maximize norm of $\xi_p R|x\rangle$ is called substate merging because equations (5) and (9) show that the final substate, $\xi_p R|x\rangle$, is a linear combination of $\xi_p|x\rangle$ and $\xi_{g(p)}|x\rangle$. This can be considered analogous to

merging $\xi_{g(p)}|x\rangle$ with $\xi_p|x\rangle$ to form a new, larger-magnitude substate $\xi_p R|x\rangle$, while some residue is left behind at $\xi_{g(p)}R|x\rangle$. For this reason, the substate merging procedure will be described as 'merging $\xi_{g(p)}|x\rangle$ into $\xi_p|x\rangle$',

$$\xi_p R|x\rangle = \sin\left(\frac{\theta}{2}\right)\xi_p g|x\rangle + e^{i\phi}\cos\left(\frac{\theta}{2}\right)\xi_p|x\rangle \tag{9}$$

$$\theta = \tan^{-1}\frac{2|\langle x|\xi_{g(p)}\xi_p|x\rangle|}{\langle x|\xi_p|x\rangle - \langle x|\xi_{g(p)}|x\rangle}$$
$$\phi = \text{phase}\left(\langle x|\xi_{g(p)}g\xi_p|x\rangle\right). \tag{10}$$

In most cases, the substate merging procedure needs to be modified such that it does not disturb the amplitude at $|\mathbf{0}\rangle$. We call this modified procedure 'controlled substate merging' and define it as follows. Given a quantum state $|x\rangle$, a pattern p, and a CX gate g that can be applied to p (and $g(p) \neq p$), the controlled substate merging procedure applies a transformation R consisting of the CX gate g and single-qubit rotations applied to the target qubit of g, such that the norm of $\xi_p R|x\rangle$ is maximized and $\langle\mathbf{0}|R|x\rangle = \langle\mathbf{0}|x\rangle$, ignoring global phase.

The substate merging procedure can be adapted for controlled substate merging as follows. First, the RZ–RY sequence R' is constructed for merging $\xi_p|x\rangle$ into $\xi_{g(p)}|x\rangle$ as shown in equation (11). In this pair of gates, only the RY gate disturbs the amplitude at $|0\rangle$, thus, the RY gate is replaced with a controlled-RY gate, then decomposed into CX and RY described in equation (12) wherein g is the CX gate given in the problem statement. The transformation R' minimizes $\xi_p R'|x\rangle$ and maximizes $\xi_{g(p)}R'|x\rangle$, whereas the opposite effect is desired. This problem is tackled by removing the trailing g in the gate sequence R' to get the desired gate sequence R in equation (13). One interesting property of this construction is that, for all integers i where $g|i\rangle = |i\rangle$, $\langle i|R|x\rangle = \langle i|x\rangle$ up to a complex phase, and the restriction $\langle\mathbf{0}|R|x\rangle = \langle\mathbf{0}|x\rangle$ is a special case of this more general property of the construction for controlled substate merging. This property will prove useful for our ISA implementation,

$$R' = RY(\theta)\,RZ(\phi). \tag{11}$$

$$\begin{aligned} R' &= CRY(\theta)\,RZ(\phi) \\ &= gRY\left(-\frac{\theta}{2}\right)gRY\left(\frac{\theta}{2}\right)RZ(\phi) \end{aligned} \tag{12}$$

$$R = RY\left(\frac{\theta}{2}\right)gRY\left(\frac{\theta}{2}\right)RZ(\phi). \tag{13}$$

2.4. Sparse QSP

In this work, we consider sparse quantum states of the form:

$$|x'\rangle = \sum_{i\in I_{p^0}} c_i|i\rangle + \sum_{j\in I_p} c_j|j\rangle \tag{14}$$

where p is a k-bit pattern, $k \leqslant 2$, p^0 is p but with its '1' characters changed to '0', and $p \neq p^0$. We also require the '$*$' characters in p to be adjacent to each other, if there are two of them, and for all pairs k and l where $p[k] = p[l] = 1$, there exists no m such that $k < m < l$ and $p[m] =$ '$*$'. Patterns meeting these latter two criteria regarding the positioning of '1' and '$*$' characters are called 'compatible', since these criteria are necessary for our sparse QSP method to work on LNN architectures specifically. For example, '$011*0$', '$100*$' and '00001' are compatible patterns, but '$01*001$' and '$00*01*$' are not. For a general non-LNN architecture, the compatibility criteria would need to be modified accordingly. Specifically, the '$*$' characters in p would not need to be adjacent, but rather, we would require them to be at positions that are connected on the connectivity graph. The remainder of the ISA implementation would then incorporate this adjusted set of compatible patterns analogously.

As a base case, if p contains at most one '1' character, and that character is adjacent to a '$*$' character, then $|x'\rangle$ is a $k+1$-qubit entangled state on neighboring qubits. In addition, $k+1 \leqslant 3$, so existing exact state preparation methods can be applied [47]; we describe our specific implementation in the appendix. Otherwise, we can construct a sequence of CX gates $g_1, g_2, \ldots, g_m$ and a sequence of patterns $p_0, p_1, \ldots, p_m$ such that $p_k = g_k(p_{k-1})$ for all $1 \leqslant k \leqslant m$, $p_0 = p$, and p_m is a base case pattern. This construction is possible

because p satisfies the compatibility criteria described above. From here, we construct the sequence $|x'_0\rangle, |x'_1\rangle, \ldots, |x'_m\rangle$ such that $|x'_k\rangle = g_k|x'_{k-1}\rangle$ and $|x'_0\rangle = |x'\rangle$. Repeatedly applying lemma A1 shows that

$$|x'_m\rangle = \sum_{i \in I_{p^0}} c'_i|i\rangle + \sum_{j \in I_{p_m}} c'_j|j\rangle \tag{15}$$

for some complex amplitudes c'_i and c'_j. Then $|x'_m\rangle$ is a $k+1$-qubit quantum state on neighboring qubits, which can be prepared by existing methods. In summary, our sparse QSP method uses a sequence of CX gates to move the p-substate to qubits adjancent to the p^0-substate, then applies exact QSP. The number of CX gates our method needs to prepare sparse quantum states is the minimum possible length m of the CX gate sequence used to move the p-substate plus the number of CX gates needed for the base case. Let $CX(p)$ denote this quantity. Breadth-first search can be used to compute $CX(p)$; despite the inefficiency of such an approach, the computation for $CX(p)$ can be precomputed and cached, and therefore, needs to be performed only once.

2.5. ISA framework implementation

In the previous sections, we described the building blocks for our ISA implementation; in this section, we assembled the pieces. For the select step, we enumerate all compatible k-bit patterns p, with $k \leqslant 2$. For each pattern, we construct the corresponding (non-normalized) sparse approximation:

$$|x'(p)\rangle = \xi_{p^0}|x\rangle + \xi_p|x\rangle. \tag{16}$$

When the sparse QSP method is applied to $|x'(p)\rangle$, the result is $\|x'(p)\||\mathbf{0}\rangle$; when the same gate sequence is applied to $|x\rangle$ to get $|x_1\rangle$, we expect $\langle\mathbf{0}|x_1\rangle = \|x'(p)\|$. This corresponds to a fidelity increase of $\langle x'(p)|x'(p)\rangle - |\langle\mathbf{0}|x\rangle|^2$ using $CX(p)$ CX gates, or a fidelity increase ratio of

$$R_p = \frac{\langle x'(p)|x'(p)\rangle - |\langle\mathbf{0}|x\rangle|^2}{1 + CX(p)}. \tag{17}$$

Adding 1 to the denominator is necessary to prevent division by zero when $CX(p)$ is zero. In the select step, we greedily choose p to maximize R_p and select the corresponding sparse approximation $|x'(p)\rangle$. In the prepare step, the sparse QSP method is adapted for general QSP. In our implementation, we replace the iterated application of CX gates to move the p-substate with the iterated application of controlled substate merging. This change allows the p-substate of $|x\rangle$ to increase its norm as it moves towards a base-case pattern, instead of keeping the same norm throughout. Also, using controlled substate merging in place of CX gates will not affect the p^0 substate, as argued at the end of section 2.3. In addition, we implement a greedy selection procedure to construct the pattern sequence $p_0, p_1, \ldots, p_m$. Specifically, we implement prepare using the following procedure:

(i) If p is a base case pattern, apply exact QSP and terminate the prepare step.
(ii) Otherwise, construct $\mathcal{P}$, the set of patterns p' such that $g(p) = p'$ for some CX gate g and $p \neq p'$.
(iii) For each $p' \in \mathcal{P}$, compute the magnitude of the substate that would result from merging $|x_p\rangle$ with $\xi_{p'}|x\rangle$ and call this quantity $\mathrm{mag}(p')$.
(iv) For each p', compute the projected fidelity increase ratio as

$$R_{p'} = \frac{\mathrm{mag}(p') + \langle x|\xi_{p^0}|x\rangle - |\langle\mathbf{0}|x\rangle|^2}{1 + \min(CX(p), CX(p'))}. \tag{18}$$

The numerator is the projected fidelity increase after substate merging, while the denominator is the projected number of CX gates—one for substate merging and $\min(CX(p), CX(p'))$ for preparing the resulting sparse approximation.
(v) Select p' such that it maximizes $R_{p'}$. If $CX(p) < CX(p')$, then controlled substate merge $\xi_{p'}|x\rangle$ into $\xi_p|x\rangle$ and return to the first step. Otherwise, controlled substate merge $\xi_p|x\rangle$ into $\xi_{p'}|x\rangle$, set $p = p'$, and return to the first step.

It may seem possible that this procedure will always select p' with $CX(p') > CX(p)$ in step 5 and enter an infinite loop. However, this cannot happen because each time another substate is merged into $\xi_p|x\rangle$, an extra CX gate is used. This extra CX gate must be justified by a corresponding increase in projected fidelity increase, and the projected fidelity increase cannot increase indefinitely. We defer the discussion of two optimizations in the ISA framework—refinement by RZ–RY and retroactive base case reduction—to appendix B. Otherwise, every step of our ISA framework implementation has been fully described, and we summarize it in figure 2.

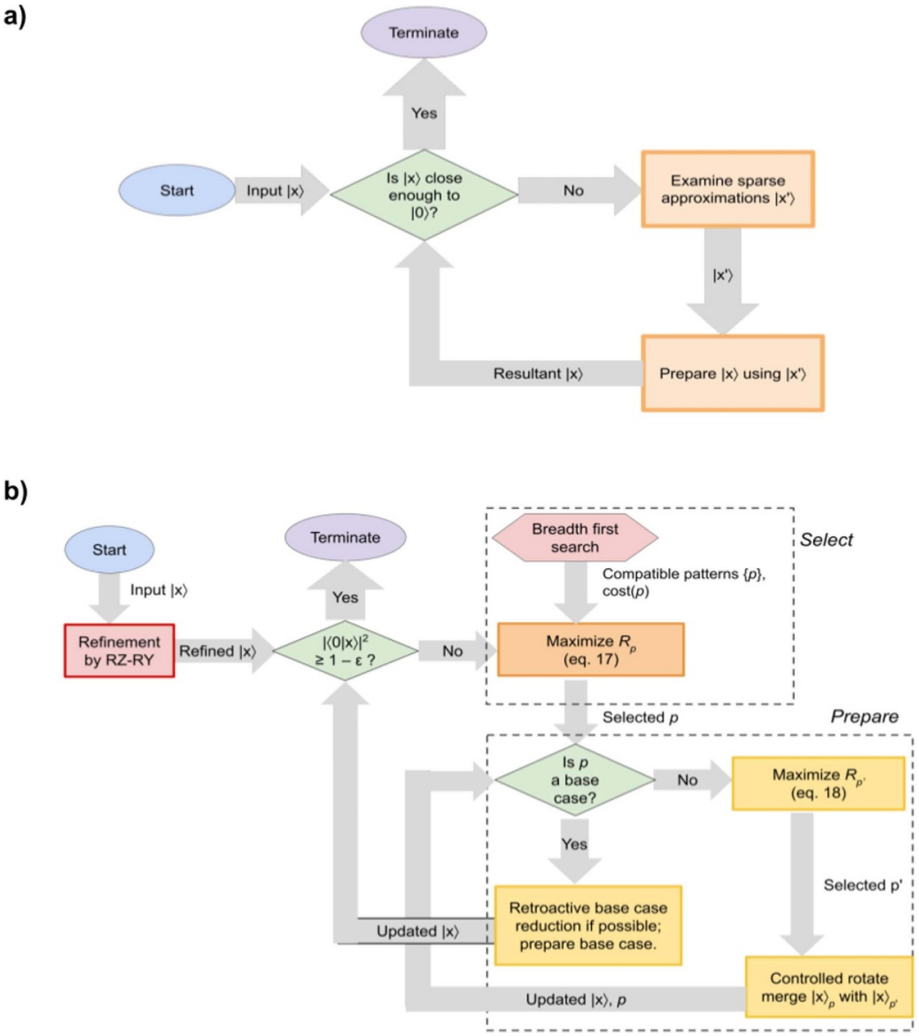

Figure 2. Schematic depiction of the ISA framework. (a) The ISA framework is a check termination-select-prepare loop that brings $|x\rangle$ closer to $|\mathbf{0}\rangle$ on every iteration. (b) Flowchart depiction of the ISA implementation which includes refinement by *RZ–RY* before entering the main check termination-select-prepare loop and retroactive base cacse reduction. Both optimizations, *RZ–RY* refinement and retroactive base case reduction, are detailed in appendix B.

3. Results and discussion

We conduct several experiments to demonstrate the creation of arbitrary quantum states through two case studies. The first case study focuses on generating uniform random states, while the second explores protein-encoded states where computed protein embeddings serve as the quantum state amplitudes. In both scenarios, the approach involves real amplitude encoding, where an n-qubit quantum state is constructed by normalizing a 2^n-dimensional feature vector and mapping its components to quantum amplitudes. The experiments utilize both the ISA method and benchmark QSP techniques to generate sequences of quantum gates that transform an initial state into the desired target state, aiming to achieve a theoretical fidelity of 0.95. To define this more precisely, given a real 2^n-dimensional vector $\mathbf{x}$ with components $x_0, x_1, \ldots, x_{2^n-1}$, the corresponding n-qubit quantum state can be expressed as:

$$|x\rangle = \sum_{j=0}^{2^n-1} \tilde{x}_j |j\rangle \tag{19}$$

where $\tilde{x}_j = \frac{x_j}{\|\mathbf{x}\|}$ represents the normalized features of $\mathbf{x}$. By inputting $|x\rangle$ into ISA and the benchmark QSP methods, we generate a gate sequence $g_1^\dagger, g_2^\dagger, \ldots, g_m^\dagger$ that prepares $|x\rangle$ from $|\mathbf{0}\rangle$ with an ideal fidelity of 0.95, as described in equation 3. It should be noted that the vector $\mathbf{x}$ corresponds to fixed length randomly sampled vector in the first case study and protein embeddings computed with a deep learning-based language model in the second case.

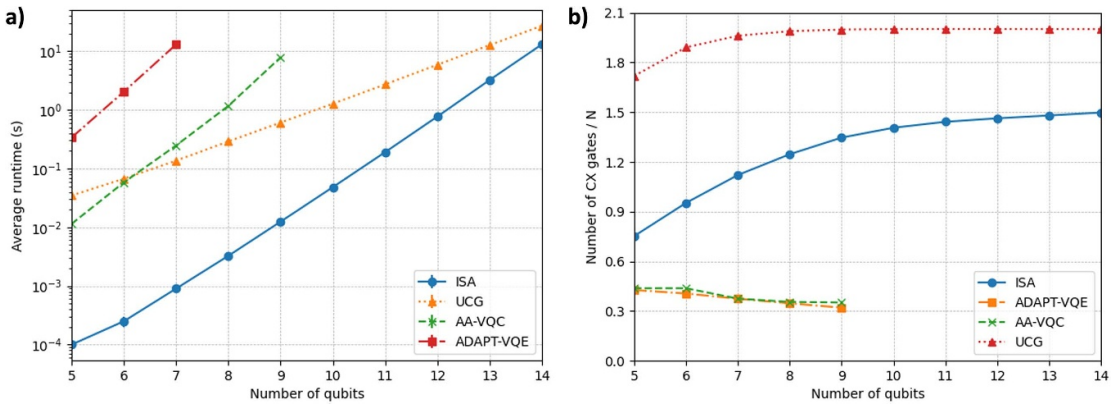

Figure 3. (a) Average classical runtime versus number of qubits for each QSP method, plotted on a linear-log scale. Only the ISA and UCG methods were able to prepare the quantum states for n up to 14; the AA-VQC method timed out (more than 1 hour to prepare 100 states) after $n = 9$ and the ADAPT-VQE method timed out after $n = 7$. (b) Average number of CX gates used to prepare 100 randomly generated quantum states versus number of qubits. The values in the ISA and ADAPT-VQE column show two decimal digits, since the number of CX gates used varied across quantum states. By contrast, the values for the UCG column were theoretically determined.

3.1. Uniform random state preparation

We first applied the proposed state preparation method to prepare uniformly sampled quantum states. Exact state preparation approaches using UCG [17], alternating ansatz VQC (AA-VQC) [48], and ADAPT-VQE [49] were also validated for the random uniform quantum states. In our experiments using randomly generated states, we tested qubit numbers from 5 to 14, inclusive. For each qubit number, 100 quantum states were uniformly, and randomly sampled. When determining the classical runtime, the number of CX gates in the AA-VQC was set to 2^{n-1}. This number was chosen because each CX gate attaches four free parameters, therefore, approximately 2^{n-1} CX gates are necessary for the number of circuit parameters to exceed the number of free parameters in an n-qubit quantum state. Under these conditions, the VQC should be able to prepare any arbitrary n-qubit quantum state with high fidelity.

To determine the number of CX gates needed for AA-VQC, we trained VQCs with varying numbers of layers and reported the CX gate count corresponding to the minimum number of layers needed to achieve an average fidelity of 0.95. However, this minimum CX count also depends on the number of gradient descent steps used to optimize the angles: more layers require less training. To ensure a sufficiently large but also fair number of training iterations for each n, we trained an AA-VQC with 2^{n-1} CX gates on the first three quantum states and recorded the number of training iterations required to achieve 0.95 fidelity on all three states. Then, the number of training iterations was set to twice the sum of those numbers. This ensured that the number of training iterations was approximately six times as many as actually necessary for that specific number of qubits, preventing insufficient training from marring the results. CX gate count experiments were not run for the UCG method. This is because the number of CX gates depends on whether the circuit transpiler finds good circuit optimizations for LNN. However, using the exact nature of this algorithm, it was determined that this method uses $2 \times 2^n + 2n - 19$ CX gates to prepare an n-qubit quantum state (proof given in the appendix).

3.2. Fidelity of protein-encoded states

For the second set of experiments, we sampled 20 proteins from the proteome of Homo sapiens from the UniProtKB database [45]. We embedded these proteins into continuous, 1024-dimensional feature vectors using the ProtT5 pre-trained protein language model [50]. We encoded these features into ten-qubit quantum states using real amplitude encoding. Classical runtime and CX gate counts were measured using the same methods as in the first set of experiments. Theoretical fidelities were also determined using the gate sequence outputs of each of the methods.

The AA-VQC method needed 71 layers to achieve a minimum average fidelity of 0.95, corresponding to 320 CX gates. The fidelity of UCG is exactly 1.0, as it is an exact state preparation method rather than an approximation. Figure 3(b) shows that the number of CX gates needed for the VQC-based methods is approximately $0.3N = 307.2$, which is consistent with the CX gate counts reported for AA-VQC and ADAPT-VQE methods in table 2. In addition, the CX gate counts for ISA and UCG in table 2 are consistent with those shown in table 1. Thus, all methods returned similar gate counts for both protein-encoded states and randomly generated states. In addition, the classical runtimes shown in table 2 for UCG and ISA methods are consistent with those presented in figure 3(a). In addition, despite the timeout for the

Table 1. Average number of *CX* gates used to prepare 100 randomly generated quantum states versus number of qubits. The values in the ISA and ADAPT-VQE column show two decimal digits, since the number of *CX* gates used varied across quantum states. By contrast, the values for the UCG column were theoretically determined, and are therefore whole numbers. In addition, the values for AA-VQC were calculated by finding the minimum number of *CX* gates corresponding to an average fidelity of 0.95 - these are also whole numbers.

Number of qubits	ISA	UCG	ADAPT-VQE	AA-VQC
5	24.08	55	13.67	14
6	60.98	121	25.97	28
7	143.46	251	47.96	48
8	319.02	509	88.76	91
9	689.32	1023	164.22	180
10	1439.6	2049	TIMEOUT	TIMEOUT
11	2952.66	4099		
12	5991.51	8197		
13	12 123.3	16 391		
14	24 538	32 777		

Table 2. Minimum *CX* gate count and classical runtime to achieve 0.95 ideal fidelity for each state preparation method for the protein-encoded quantum states. Each result is an average across the 20 trials of preparing distinct protein-encoded quantum states.

Method	*CX* gate count	Classical runtime
ISA	1422.3	6.420×10^{-2}
UCG	2049	1.801×10^{0}
ADAPT-VQE	286.0	6.424×10^{3}
AA-VQC	320	4.895×10^{1}

VQC-based methods at ten qubits, as shown in figure 3, the classical runtime values are consistent with the general trend of pure VQC being several orders of magnitude slower than ISA and ADAPT-VQE being an order of magnitude slower than AA-VQC. Thus, the classical runtime results for protein-encoded states are consistent with those for general quantum states.

In addition, we also benchmark the proposed ISA approach against MPS for preparing quantum states corresponding to protein-encoded states. The computational metrics presented in appendix D reveal significant differences in their practical applicability for QSP, particularly for highly entangled protein-encoded states. It is clear that the performance of MPS is strongly dictated by its bond dimension hyperparameter. As the bond dimension increases, the average *CX* gate count for 10-qubit quantum states increases significantly as compared to the ISA approach. It should also be noted that the computational efforts also increase significantly without an improvement in the achieved fidelity. In contrast, the ISA method's iterative approach offers a more practical solution, as it naturally progresses toward the target fidelity without requiring extensive hyperparameter optimization. This fundamental difference makes ISA more suitable for preparing highly entangled states, where the MPS approach struggles to capture the complex quantum correlations despite increased computational resources. The requirement for hyperparameter tuning in MPS adds an extra layer of complexity and uncertainty to the state preparation process, making it less practical for real-world applications involving protein-encoded states.

3.3. Computational performance

Figure 3(a) shows the average classical runtime versus the number of qubits for each method. For all numbers of qubits tested, the ISA implementation ran the fastest, however, the ISA method's classical runtime increases faster with qubit number compared to the UCG method. Thus, the UCG method is expected to be faster than ISA for more than 14 qubits. Both methods were orders of magnitude faster than the variational methods. This reflects the intense computational cost incurred by the gate optimization procedures.

Table 1 shows the average number of *CX* gates used by each method as a function of the number of qubits. The UCG method uses the most *CX* gates. The ISA method uses somewhat fewer *CX* gates, while the variational methods use the fewest *CX* gates. The variational methods far outperform the non-variational methods in this regard, demonstrating the effectiveness of the variational optimization procedure. That said, the ISA method is shown to outperform the UCG method without any variational angle tuning procedure, indicating that ISA is able to find good QSP gate sequences.

The number of free parameters in the quantum circuit is proportional to the number of *CX* gates; the number of free parameters needed is proportional to N, the number of complex amplitudes in the target state. Therefore, we hypothesized that the number of *CX* gates would be approximately proportional to N and try to determine the constant factor by graphing *CX* count divided by N as a function of the number of

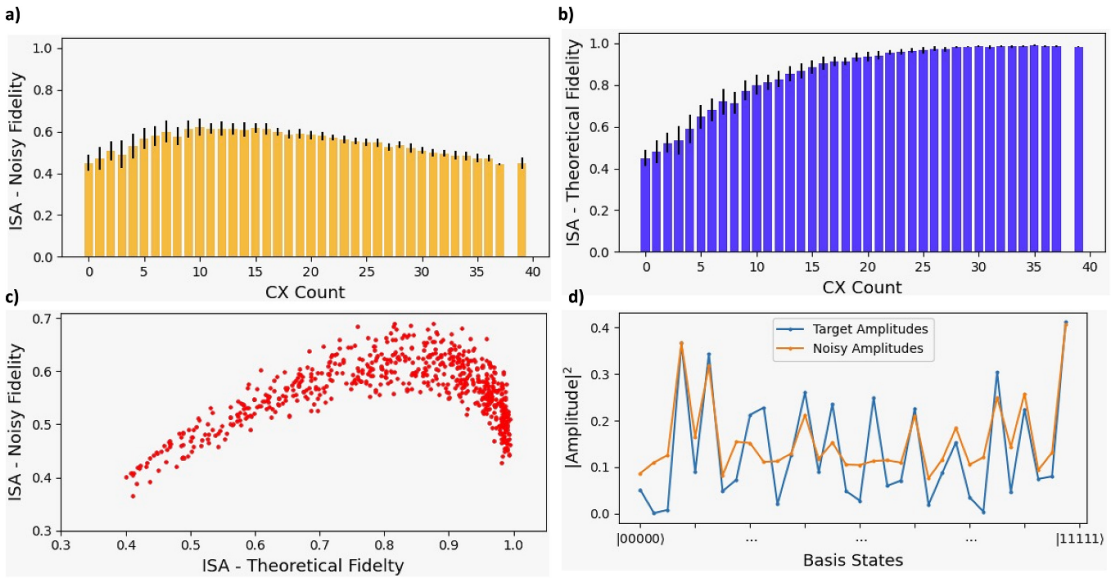

Figure 4. Theoretical fidelity, noisy fidelity, and *CX* gate count for ISA on randomly sampled five-qubit states on imbq_melbourne quantum hardware. (a)Noisy fidelity versus *CX* gate count across all 100 states. The error bars represent one standard deviation at each *CX* gate count. The bar chart shows that the noisy fidelity peaks around 10 *CX* gates. (b) Theoretical fidelity versus *CX* gate count across all 100 states. The error bars represent one standard deviation at each *CX* count. The graph shows that as *CX* gate count increases, theoretical fidelity increases. (c) Noisy fidelity versus theoretical fidelity. The scatter plot shows that increasing theoretical fidelity increases the noisy fidelity up to a theoretical fidelity of 0.8, then the curve inverts and the two quantities become negatively correlated. (d) Target state and prepared state amplitude vs basis vector for the first randomly sampled quantum state. This figure compares the target quantum state with the quantum state prepared on noisy hardware—while the peaks, corresponding to large amplitudes, approximately match, the valleys corresponding to smaller amplitudes match less well.

qubits. If the *CX* count is indeed proportional to N, then the graph should approach a horizontal line for large n, asymptotically approaching the constant factor. Figure 3(b) shows the average number of *CX* gates divided by N as a function of qubit number for each method. For our ISA implementation, the curve increases for $n < 10$, then stabilizes to around 1.5. Thus, we conclude that our ISA implementation uses approximately $1.5N$ *CX* gates for randomly sampled quantum states. By contrast, the UCG method is shown to use $2N + 2n - 19$ *CX* gates, which is $2N$ to the leading order. Thus, the ISA method is shown to use approximately 25% fewer *CX* gates compared to the UCG method. However, both variational methods stabilize to around 0.3, which is about 5 times better than our ISA implementation, demonstrating the effectiveness of the (computationally expensive) gate optimization procedure.

3.4. Simulations with noisy quantum circuits

For the third set of experiments, we prepared five-qubit quantum states on Qiskit's FakeMelbourneV2 machine, which features a LNN architecture on the first five qubits. We sampled 100 five-qubit states uniformly at random. For each state, we ran our ISA implementation with target fidelity thresholds ranging from 0.4 to 0.95 in increments of 0.05, then included an additional target fidelity threshold of 0.98, for a total of 13 trials per state. For each trial, the theoretical fidelity, noisy fidelity, and *CX* gate count were determined. Different target fidelity thresholds for the same state sometimes yielded identical gate sequences and corresponding fidelities; these duplicates were removed from our results.

Figure 4 depicts the theoretical fidelity, noisy fidelity, and *CX* gate count from the third set of experiments. Figure 4 shows that as the theoretical fidelity increases from 0.4 to 0.8, the noisy fidelity increases from 0.4 to 0.6. Further increases in theoretical fidelity, however, cause the noisy fidelity to decrease. This inverse-U shaped correlation between the noisy fidelity and the theoretical fidelity is expected because increasing target fidelity increases the *CX* gate count; at first, *CX* gates improve the quality of the prepared state, but beyond a point, the marginal utility of a *CX* gate is outweighed by the noise it introduces. Indeed, figures 4(a) and (b) corroborate these expectations; while increasing *CX* gate count monotonically increases theoretical fidelity, an increase in *CX* gate count improves noisy fidelity only up to a peak of 0.68, around 10 *CX* gates. Figure 4(d) shows that the largest amplitudes in the prepared states match the largest amplitudes in the target state, but the smaller amplitudes match less well. This is a direct consequence of how our ISA implementation approximates the target state as the sum of its largest amplitudes, then applies

corrections for the smaller amplitudes. These corrections cannot prepare the smaller amplitudes perfectly, leading to a negative correlation between target amplitude magnitude and how closely it matches the corresponding amplitude in the prepared state.

3.5. Hardware considerations and challenges

For both the random and protein-encoded quantum states, our implementation of the ISA framework uses fewer *CX* gates than exact state preparation methods without incurring the variational optimization costs associated with VQC-based methods. However, the VQC-based methods are able to prepare quantum states using much fewer *CX* gates compared to the ISA method, and the UCG method is expected to be faster than ISA for quantum states on more than 14 qubits. This trend is expected: the less computation time a method uses, the lower the quality of its output.

In practice, every *CX* gate applied introduces hardware noise into the computation. Therefore, the design of quantum circuits will require a careful balance between using enough *CX* gates to approximate the desired computation, while not using so many *CX* gates as to corrupt the computation with noise. The ISA framework lends itself to a simple method for finding this balance: if too many *CX* gates are present in the state preparation algorithm, delete the gates corresponding to the last ISA iteration; if insufficient *CX* gates are present, perform more sparse approximation iterations, which lends itself favorable for near-term quantum devices. This works because each iteration of sparse approximation monotonically increases the state preparation fidelity and each additional iteration provides diminishing returns on fidelity increase. This idea was demonstrated our noisy simulation experiments. Different target fidelity thresholds led to different numbers of *CX* gates, corresponding to a range of different observed preparation fidelities. Finding the optimal *CX* count was reduced to identifying the optimal target fidelity threshold along the theoretical versus noisy fidelity plot for that particular state. By contrast, creating such a *CX* gate count vs noisy fidelity curve for a simple VQC method would require re-training the quantum circuit at every *CX* count sampled; for large numbers of qubits, this would require many repetitions of the parameter optimization procedure, driving up its already-expensive computational cost.

It would be valuable for future research to explore how ISA could integrate error correction techniques or noise mitigation strategies. For instance, ISA could be combined with noise mitigation methods such as zero-noise extrapolation [51, 52] or probabilistic error cancellation [51, 53], which enhance fidelity during post-processing without modifying the gate sequence. Additionally, it has been shown that repeat-until-success paradigms can enhance state preparation algorithms, including potentially ISA, by providing additional robustness against hardware noise [54]. This is achieved through the use of mid-circuit partial measurements, which allow verification of whether a particular operation was successfully applied [55]. Furthermore, incorporating quantum error correction codes, such as qLDPC codes [56] or surface codes [57, 58], into ISA could enable the resulting gate sequence to be made fault-tolerant. This can be achieved using established methods for decomposing arbitrary rotations into fault-tolerant gate sets [59–61]. Periodic syndrome measurements during ISA iterations could detect and correct errors, thereby preserving high-fidelity state preparation.

In addition, different quantum hardwares have different qubit connectivity. The only hardware-dependent steps in our ISA implementation are enumerating the compatible patterns $\{p\}$ and computing $CX(p)$, both of which can be done by an analogous breadth first search procedure. Thus, our ISA implementation can be easily adapted to different hardware connectivities. However, the ISA method does face several limitations. Most importantly, our implementation of ISA uses four or five times as many *CX* gates as the VQC-based methods. This may cause ISA-generated gate sequences to induce much more noise on near term quantum hardware compared to VQC-based methods. This problem may eventually be mitigated by future work discovering alternative, easy-to-prepare classes of quantum states for approximating dense quantum states, then adapting those discoveries towards an improved ISA implementation. In addition, it may be possible to incorporate some variational training alongside ISA. However, the cost function, training schedule, gradient conditioning, and other things would need to be carefully designed to ensure such a hybrid method does not become a worse version of ADAPT-VQE or AA-VQC. Another limitation of the ISA method is that it cannot prepare quantum states that require extremely high fidelity. This is because the ISA method treats the dense target state as a sparse approximate state, therefore, some percentage of the fidelity will be lost on every sparse approximation, and chasing down these lost details would require many, many more iterations of ISA. That said, many applications of QSP, such as machine learning and sampling from probability distributions, require only approximate QSP [23]. Nevertheless, in this context of approximate QSP, ISA can provide an advantage because the fidelity threshold-and consequently the circuit depth-can be adjusted according to the specific requirements of the downstream machine learning task wherein the corresponding metrics can guide the search for optimal ISA iterations as a hyperparameter.

4. Conclusion

In this work, we introduced a framework for efficient QSP that addresses key challenges in encoding chemical data like proteins on quantum computers. The proposed ISA framework demonstrated several important advantages while balancing the competing demands of gate count optimization, classical computational efficiency, and hardware constraints. The ISA framework achieves approximate QSP using approximately 25% fewer *CX* gates compared to exact methods like UCG, while maintaining compatibility with LNN architectures common in current quantum hardware. This reduction in gate count is crucial for minimizing decoherence effects in near-term quantum devices. The presented results also demonstrated that ISA can prepare quantum states orders of magnitude faster than variational approaches, making it particularly suitable for large-scale applications in protein data encoding. The framework successfully prepared both random uniform quantum states up to 14 qubits and protein-encoded quantum states from the UniProtKB database with high fidelity. The method demonstrated consistent performance across different state preparation tasks, suggesting robust applicability to various quantum computing applications. The limitations of our approach include higher gate counts compared to variational methods and potential challenges in achieving extremely high fidelities. However, these limitations are offset by the method's classical computational efficiency and hardware adaptability. Future work could explore alternative sparse approximation strategies or hybrid approaches incorporating limited variational optimization to further improve gate count efficiency while maintaining the framework's computational advantages.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

The data, source code, and results that support the findings of this study are openly available in our GitHub repository at https://github.com/PEESEgroup/QSP-ISA-Protein. This includes Python and C++ code comprising implementation of all quantum state preparation algorithms described in this work, input data files containing protein-encoded quantum states, and complete benchmark results included in the log files.

Appendix A

Lemma A1. If an *X* or *CX* gate g may be applied to a pattern p, then for all quantum states $|x\rangle$, $g\xi_p|x\rangle = \xi_{g(p)}g|x\rangle$, $\forall g \in \{X(i), CX(i,j)|p_i \neq *, p_j \neq *\}$

Proof. Define $g(i)$ such that $g|i\rangle = |g(i)\rangle$. Then $g(i) \in I_{g(p)}$ for all $i \in I_p$. This also implies $g(g(i)) \in I_{g(g(p))}$ for all $g(i) \in I_{g(p)}$. Applying g twice amounts to the identity transformation in the context of both patterns and substates, thus, $i \in I_p$ for all $g(i) \in I_{g(p)}$, implying that g creates a bijection between I_p and $I_{g(p)}$. In addition, given that *X* and *CX* gates on quantum states represent amplitude permutations in the computational basis, $\langle g(i)|g|x\rangle = \langle i|x\rangle$. It then follows that,

$$\begin{aligned} g\xi_p|x\rangle &= g\sum_{i\in I_p}|i\rangle\langle i|x\rangle \\ &= \sum_{i\in I_p}|g(i)\rangle\langle i|x\rangle \\ &= \sum_{i\in I_p}|g(i)\rangle\langle g(i)|g|x\rangle \\ &= \sum_{j\in I_{g(p)}}|j\rangle\langle j|g|x\rangle \\ &= \xi_{g(p)}g|x\rangle. \end{aligned} \tag{20}$$

□

Appendix B. Refinements for the ISA method

We describe two optimizations to our implementation of the ISA framework: refinement by *RZ–RY*, and retroactive base case reduction.

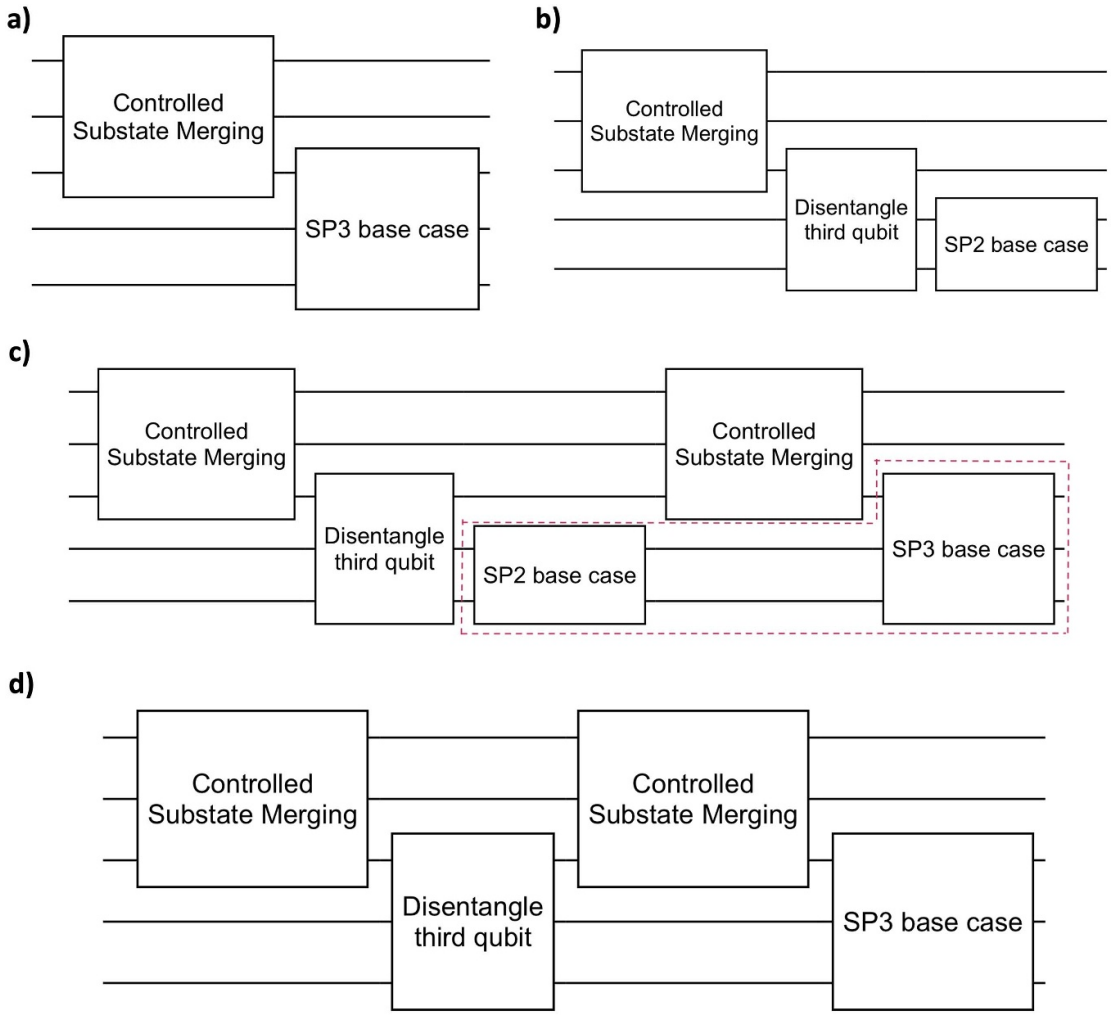

Figure 5. Quantum circuit components comprising the ISA framework. (a) General quantum circuit structure of a single iteration of ISA using a two-bit pattern p. Several iterations of controlled substate merging are performed on neighboring qubits before a three-qubit state preparation base case is applied. (b) Decomposition of a single iteration of ISA using a two-bit pattern p. The three-qubit state preparation base case is broken down into a distangling step and a two-qubit state preparation base case. (c) General quantum circuit structure for two consecutive iterations of ISA ending in a three-qubit base case on the same three qubits. Note that the SP2 base case in the first ISA iteration can be commuted with the controlled substate merging step in the second ISA iteration. (d) General quantum circuit structure for two consecutive iterations of ISA width the same three-qubit state preparation base case after applying retroactive base case reduction. The SP2 block from figure 4 is merged with the later SP3 block, saving 1 CX gate.

B.1. Refinement by *RZ–RY*

Refinement by *RZ–RY* seeks to apply approximate QSP to the target state $|x\rangle$ without using any CX gates, before applying the ISA framework. This is done by selecting a zero-bit pattern p such that $\|\xi_p|x\rangle\|$ is maximized (ties are broken randomly), then applying substate merging several times to eventually merge $\xi_p|x\rangle$ into $\xi_{p'}|x\rangle$. The selection procedure resembles the implementation of prepare, but using substate merging instead of controlled substate merging:

(i) If p is the string of all zeroes, then terminate this procedure. Otherwise, continue.
(ii) Construct the set P, containing zero-bit patterns p' that differ from p.
(iii) Select p' such that $\xi_{p'}|x\rangle$ is maximized, with ties broken randomly.
(iv) Let i be the index where p and p' differ. If $p[i] = 0$ then merge $\xi_{p'}|x\rangle$ into $\xi_p|x\rangle$, otherwise, merge $\xi_p|x\rangle$ into $\xi_{p'}|x\rangle$ and update $p = p'$. Return to step 1.

B.2. Retroactive Base Case Reduction

When a two-bit pattern p is chosen in the select step, the corresponding quantum circuit takes on the structure shown in figure 5(a). The three-qubit QSP step can be broken into two steps: disentangling the third qubit and two-qubit state preparation on the remaining two qubits, as shown in Figure 5(b). However,

if two consecutive iterations of ISA both selected a two-bit pattern and those two patterns had their '$*$' characters on the same two qubits, the resulting quantum circuit would have the structure shown in figure 5(c). Then, the part of the quantum circuit enclosed in dotted lines first performs two-qubit state preparation on the lower two qubits, then performs three-qubit state preparation on all three relevant qubits. These two transformations have the net effect of transforming the three-qubit substate to $|\mathbf{0}\rangle$ and can instead be replaced by a single three-qubit state preparation routine. This procedure saves the one CX gate used by the two-qubit state preparation base case.

Appendix C. Optimal, exact three-qubit state preparation

Optimal, exact three-qubit state preparation has been studied in the literature [47]; our specific implementation requires three steps, called 'D1', 'D2' and 'SP2'. Let the target state be

$$|x\rangle = \sum_{i,j,k\in\{0,1\}} a_{ijk}|ijk\rangle. \tag{21}$$

Step D1 applies a single-qubit rotation, then $CX(0, 1)$, then another single-qubit rotation, resulting in the state

$$|y\rangle = \sum_{i,j,k\in\{0,1\}} b_{ijk}|ijk\rangle, \tag{22}$$

such that

$$\frac{b_{110}}{b_{010}} = \frac{b_{111}}{b_{011}} \quad and \quad \frac{b_{100}}{b_{000}} = \frac{b_{101}}{b_{001}}. \tag{23}$$

Next, step D2 uses a single-qubit rotation, $CX(1, 2)$, and another single-qubit rotation to transform $|y\rangle$ into a two-qubit state

$$|z\rangle = \sum_{j,k\in\{0,1\}} c_{jk}|0jk\rangle. \tag{24}$$

Finally, step SP2 implements two-qubit state preparation on this state using one CX gate and several single-qubit rotations. We describe each of these steps in detail, starting with SP2 and D2. Then, we briefly discuss the decomposition of uniformly controlled single-qubit rotations before concluding with a description of D1.

C.1. SP2

Starting from the state

$$|z\rangle = \sum_{j,k\in\{0,1\}} c_{jk}|0jk\rangle, \tag{25}$$

we rotate merge $c_{001}|001\rangle$ into $c_{000}|000\rangle$, leading to the state:

$$|z'\rangle = c'_{000}|000\rangle + c'_{010}|010\rangle + c'_{011}|011\rangle. \tag{26}$$

Next, $c'_{011}|011\rangle$ is controlled rotate merged into $c'_{010}|010\rangle$. This results in a one-qubit state that can be prepared using a single-qubit rotation gate.

C.2. Step D2

Starting from the state

$$|y\rangle = \sum_{i,j,k\in\{0,1\}} b_{ijk}|ijk\rangle, \tag{27}$$

we rotate merge $b_{100}|100\rangle$ into $b_{000}|000\rangle$. The pre-condition

$$\frac{b_{100}}{b_{000}} = \frac{b_{101}}{b_{001}} \tag{28}$$

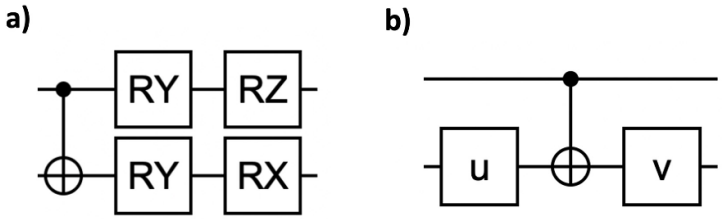

Figure 6. (a) The circuit block used in ADAPT-VQE for state preparation. The operator pool was generated by translating this block across different pairs of adjacent qubits. Each operator contains four parameters, one for each rotation. All four rotation angles were initialized to zero. (b) Target decomposition of the uniformly controlled single-qubit rotation with one control qubit.

ensures that this procedure also merges $b_{101}|101\rangle$ into $b_{001}|001\rangle$. This results in the state:

$$|y\rangle = \sum_{i,j,k\in\{0,1\}} b'_{ijk}|ijk\rangle$$
$$\text{such that} \quad b'_{100}, b'_{101} = 0. \tag{29}$$

In addition, the *RZ* and *RY* gate were applied to qubit 2, therefore, the equality

$$\frac{b_{110}}{b_{010}} = \frac{b_{111}}{b_{011}} \tag{30}$$

is preserved after the transformation.

$$\frac{b'_{110}}{b'_{010}} = \frac{b'_{111}}{b'_{011}}. \tag{31}$$

Next, $b'_{110}|110\rangle$ is controlled rotate merged into $b'_{010}|010\rangle$. The equality

$$\frac{b'_{110}}{b'_{010}} = \frac{b'_{111}}{b'_{011}} \tag{32}$$

ensures that $b'_{111}|111\rangle$ is also merged into $b'_{011}|011\rangle$ by this process. The previous step zeroed out the $|100\rangle$ and $|101\rangle$ basis vectors; this step zeroed out the $|110\rangle$ and $|111\rangle$ basis vectors. Therefore, the result of these transformations is a two-qubit state:

$$|z\rangle = \sum_{j,k\in\{0,1\}} c_{jk}|0jk\rangle. \tag{33}$$

C.3. Decomposing uniformly controlled single-qubit rotations

In this section, we describe the set of uniformly controlled single-qubit rotations with one control qubit that can be decomposed into one *CX* gate and single-qubit rotations. In addition, we give a procedure for their decomposition. These constructions will be necessary for the construction of step D1 for optimal, exact three-qubit state preparation.

For simplicity, let the control qubit be qubit 1, let the target qubit be qubit 0. Suppose the desired gate applies a if the control qubit is 0, applies b otherwise. Then the desired gate has matrix form

$$\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}. \tag{34}$$

We wish to decompose this transformation into a quantum circuit of the form shown in figure 6(b).

These gates, together, take the matrix form

$$\begin{bmatrix} v & 0 \\ 0 & v \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix} \begin{bmatrix} u & 0 \\ 0 & u \end{bmatrix}. \tag{35}$$

This requires some u and v to exist such that $a = vu$ and $b = vXu$. Then

$$ab^\dagger = (vu)(vXu)^\dagger = vXv^\dagger, \tag{36}$$

which implies that $ab^\dagger$ must have the same eigenvalues, ± 1, as X. This condition is both necessary and sufficient for the uniformly controlled rotation to be decomposable into the quantum circuit shown. Indeed, if $ab^\dagger$ has the same eigenvalues as X, then we can compute v by diagonalization, then compute $u = v^\dagger a$. Finally, u and v can be decomposed to RY and RZ gates using ZYZ decomposition.

C.4. Step D1

Step D1 is implemented using a uniformly controlled single-qubit rotation, with qubit 0 as the control, qubit 1 as the target. To implement D1, we construct single-qubit operations a and b such that applying $\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$ to $|x\rangle$ leads to the state

$$|y\rangle = \sum_{i,j,k\in\{0,1\}} b_{ijk}|ijk\rangle, \tag{37}$$

such that

$$\frac{b_{110}}{b_{010}} = \frac{b_{111}}{b_{011}} \text{and} \frac{b_{100}}{b_{000}} = \frac{b_{101}}{b_{001}}. \tag{38}$$

In addition, we perform this construction such that $ab^\dagger$ has eigenvalues ± 1 to ensure the transformation can be implemented using only one CX gate.

We parameterize a and b as:

$$\begin{aligned} a &= e^{i\phi_g}\begin{bmatrix} \cos(\theta_a) & -\sin(\theta_a)e^{i\phi_a} \\ \sin(\theta_a)e^{i\alpha_a} & \cos(\theta_a)e^{i(\alpha_a+\phi_a)} \end{bmatrix} \\ b &= \begin{bmatrix} \cos(\theta_b) & -\sin(\theta_b)e^{i\phi_b} \\ \sin(\theta_b)e^{i\alpha_b} & \cos(\theta_b)e^{i(\alpha_b+\phi_b)} \end{bmatrix}. \end{aligned} \tag{39}$$

Then, we can express the b_{ijk} coefficients of $|y\rangle$ in terms of ϕ_g, θ_a, α_a, ϕ_a, θ_b, α_b, ϕ_b, and the a_{ijk} coefficients of $|x\rangle$:

$$\begin{aligned} b_{000} &= e^{i\phi_g}\left(\cos(\theta_a)a_{000} + \sin(\theta_a)e^{i\phi_a}a_{010}\right) \\ b_{010} &= e^{i(\alpha_a+\phi_g)}\left(-\sin(\theta_a)a_{000} + \cos(\theta_a)e^{i\phi_a}a_{010}\right) \\ b_{100} &= e^{i\phi_g}\left(\cos(\theta_a)a_{100} + \sin(\theta_a)e^{i\phi_a}a_{110}\right) \\ b_{110} &= e^{i(\alpha_a+\phi_g)}\left(-\sin(\theta_a)a_{100} + \cos(\theta_a)e^{i\phi_a}a_{110}\right) \\ b_{001} &= \cos(\theta_b)a_{001} + \sin(\theta_b)e^{i\phi_b}a_{011} \\ b_{011} &= e^{i\alpha_b}\left(-\sin(\theta_b)a_{001} + \cos(\theta_b)e^{i\phi_b}a_{011}\right) \\ b_{101} &= \cos(\theta_b)a_{101} + \sin(\theta_b)e^{i\phi_b}a_{111} \\ b_{111} &= e^{i\alpha_b}\left(-\sin(\theta_b)a_{101} + \cos(\theta_b)e^{i\phi_b}a_{111}\right). \end{aligned} \tag{40}$$

Then the constraints

$$\frac{b_{110}}{b_{010}} = \frac{b_{111}}{b_{011}} \text{and} \frac{b_{100}}{b_{000}} = \frac{b_{101}}{b_{001}}. \tag{41}$$

Can be written as:

$$\begin{aligned} \frac{e^{i(\alpha_a+\phi_g)}\left(-\sin(\theta_a)a_{100} + \cos(\theta_a)e^{i\phi_a}a_{110}\right)}{e^{i(\alpha_a+\phi_g)}\left(-\sin(\theta_a)a_{000} + \cos(\theta_a)e^{i\phi_a}a_{010}\right)} &= \frac{e^{i\alpha_b}\left(-\sin(\theta_b)a_{101} + \cos(\theta_b)e^{i\phi_b}a_{111}\right)}{e^{i\alpha_b}\left(-\sin(\theta_b)a_{001} + \cos(\theta_b)e^{i\phi_b}a_{011}\right)} \\ \frac{e^{i\phi_g}\left(\cos(\theta_a)a_{100} + \sin(\theta_a)e^{i\phi_a}a_{110}\right)}{e^{i\phi_g}\left(\cos(\theta_a)a_{000} + \sin(\theta_a)e^{i\phi_a}a_{010}\right)} &= \frac{\cos(\theta_b)a_{101} + \sin(\theta_b)e^{i\phi_b}a_{111}}{\cos(\theta_b)a_{001} + \sin(\theta_b)e^{i\phi_b}a_{011}}. \end{aligned} \tag{42}$$

The $e^{i\alpha_a}$, $e^{i\alpha_b}$, $e^{i\phi_g}$ terms cancel out, making ϕ_g, α_a, and α_b free parameters. Multiplying out the denominators, dividing by $\cos(\theta_a)\cos(\theta_b)$, and re-arranging the terms gives:

$$\begin{aligned} A\tan(\theta_a)\tan(\theta_b) + B\tan(\theta_a)e^{i\phi_b} + C\tan(\theta_b)e^{i\phi_a} + De^{i(\phi_a+\phi_b)} &= 0 \\ A - B\tan(\theta_b)e^{i\phi_b} - C\tan(\theta_a)e^{i\phi_a} + D\tan(\theta_a)\tan(\theta_b)e^{i(\phi_a+\phi_b)} &= 0 \end{aligned} \tag{43}$$

where

$$\begin{aligned} A &= a_{100}a_{001} - a_{000}a_{101} \\ B &= a_{100}a_{011} - a_{000}a_{111} \\ C &= a_{110}a_{001} - a_{010}a_{101} \\ D &= a_{110}a_{011} - a_{010}a_{111}. \end{aligned} \tag{44}$$

Dividing the first equation by $e^{i(\phi_a+\phi_b)}$ and taking its conjugate results in the following system of equations:

$$\begin{aligned} A^*\tan(\theta_a)\tan(\theta_b)e^{i(\phi_a+\phi_b)} + B^*\tan(\theta_a)e^{i\phi_a} + C^*\tan(\theta_b)e^{i\phi_b} + D^* = 0 \\ A - B\tan(\theta_b)e^{i\phi_b} - C\tan(\theta_a)e^{i\phi_a} + D\tan(\theta_a)\tan(\theta_b)e^{i(\phi_a+\phi_b)} = 0. \end{aligned} \tag{45}$$

Performing the substitution

$$\begin{aligned} x &= \tan(\theta_a)e^{i\phi_a} \\ y &= \tan(\theta_b)e^{i\phi_b}. \end{aligned} \tag{46}$$

Results in the system of equations:

$$\begin{aligned} A^*xy + B^*x + C^*y + D^* = 0 \\ A - By - Cx + Dxy = 0. \end{aligned} \tag{47}$$

Which can be solved by substitution. Once x and y are known, θ_a, θ_b, ϕ_a, ϕ_b can be determined.

Next, the constraint on the eigenvalues of $ab^\dagger$ needs to be satisfied. Using the same parameterization as before, we can write:

$$\begin{aligned} a &= e^{i\phi_g}\begin{bmatrix}\cos(\theta_a) & -\sin(\theta_a)e^{i\phi_a} \\ \sin(\theta_a)e^{i\alpha_a} & \cos(\theta_a)e^{i(\phi_a+\alpha_a)}\end{bmatrix} = e^{i\phi_g}\begin{bmatrix}1 & 0 \\ 0 & e^{i\alpha_a}\end{bmatrix}\begin{bmatrix}\cos(\theta_a) & -\sin(\theta_a)e^{i\phi_a} \\ \sin(\theta_a) & \cos(\theta_a)e^{i\phi_a}\end{bmatrix} \\ b &= \begin{bmatrix}\cos(\theta_b) & -\sin(\theta_b)e^{i\phi_b} \\ \sin(\theta_b)e^{i\alpha_b} & \cos(\theta_b)e^{i(\phi_b+\alpha_b)}\end{bmatrix} = \begin{bmatrix}1 & 0 \\ 0 & e^{i\alpha_b}\end{bmatrix}\begin{bmatrix}\cos(\theta_b) & -\sin(\theta_b)e^{i\phi_b} \\ \sin(\theta_b) & \cos(\theta_b)e^{i\phi_b}\end{bmatrix} \\ ab^\dagger &= e^{i\phi_g}\begin{bmatrix}1 & 0 \\ 0 & e^{i\alpha_a}\end{bmatrix}\left(\begin{bmatrix}\cos(\theta_a) & -\sin(\theta_a)e^{i\phi_a} \\ \sin(\theta_a) & \cos(\theta_a)e^{i\phi_a}\end{bmatrix}\begin{bmatrix}\cos(\theta_b) & -\sin(\theta_b)e^{i\phi_b} \\ \sin(\theta_b) & \cos(\theta_b)e^{i\phi_b}\end{bmatrix}^\dagger\right)\begin{bmatrix}1 & 0 \\ 0 & e^{i\alpha_b}\end{bmatrix}^\dagger. \end{aligned} \tag{48}$$

The two matrices between the parentheses in equation (48) are unitary matrices, thus, their product is a unitary matrix and can be expressed as

$$\begin{bmatrix}\cos(\theta_a) & -\sin(\theta_a)e^{i\phi_a} \\ \sin(\theta_a) & \cos(\theta_a)e^{i\phi_a}\end{bmatrix}\begin{bmatrix}\cos(\theta_a) & -\sin(\theta_a)e^{i\phi_a} \\ \sin(\theta_a) & \cos(\theta_a)e^{i\phi_a}\end{bmatrix}^\dagger = e^{i\alpha_g}\begin{bmatrix}\cos(\theta) & -\sin(\theta)e^{i\phi} \\ \sin(\theta)e^{i\alpha} & \cos(\theta)e^{i(\alpha+\phi)}\end{bmatrix} \tag{49}$$

for some α_g, α, ϕ, and θ. Then, we can write

$$\begin{aligned} ab^\dagger &= e^{i\phi_g}\begin{bmatrix}1 & 0 \\ 0 & e^{i\alpha_a}\end{bmatrix}e^{i\alpha_g}\begin{bmatrix}\cos(\theta) & -\sin(\theta)e^{i\phi} \\ \sin(\theta)e^{i\alpha} & \cos(\theta)e^{i(\alpha+\phi)}\end{bmatrix}\begin{bmatrix}1 & 0 \\ 0 & e^{i\alpha_b}\end{bmatrix}^\dagger \\ &= e^{i(\phi_g+\alpha_g)}\begin{bmatrix}\cos(\theta) & -\sin(\theta)e^{i(\phi-\alpha_b)} \\ \sin(\theta)e^{i(\alpha+\alpha_a)} & \cos(\theta)e^{i(\phi+\alpha+\alpha_a-\alpha_b)}\end{bmatrix}. \end{aligned} \tag{50}$$

The characteristic polynomial of this matrix is:

$$\begin{aligned} p(t) &= \left(\cos(\theta)e^{i(\phi_g+\alpha_g)} - t\right)\left(\cos(\theta)e^{i(\phi+\alpha+\alpha_a-\alpha_b+\phi_g+\alpha_g)} - t\right) \\ &\quad - \left(-\sin(\theta)e^{i(\phi-\alpha_b+\phi_g+\alpha_g)}\right)\left(\sin(\theta)e^{i(\alpha-\alpha_a+\phi_g+\alpha_g)}\right) \\ &= t^2 - e^{i(\phi_g+\alpha_g)}\cos(\theta)\left(1+e^{i(\phi+\alpha+\alpha_a-\alpha_b)}\right)t + \cos(\theta)^2 e^{i\left(2(\phi_g+\alpha_g)+\phi+\alpha+\alpha_a-\alpha_b\right)} \\ &\quad + \sin(\theta)^2 e^{i\left(2(\phi_g+\alpha_g)+\phi+\alpha+\alpha_a-\alpha_b\right)} \\ &= t^2 - e^{i(\phi_g+\alpha_g)}\cos(\theta)(1+e^{i(\phi+\alpha+\alpha_a-\alpha_b)})t + e^{2i(\phi_g+\alpha_g)}e^{i(\phi+\alpha+\alpha_a-\alpha_b)}. \end{aligned} \tag{51}$$

We require the eigenvalues to be ± 1, so the characteristic polynomial must be equal to $t^2 - 1$. This requires:

$$\begin{aligned} e^{i(\phi_g+\alpha_g)}\cos(\theta)\left(1+e^{i(\phi+\alpha+\alpha_a-\alpha_b)}\right) &= 0 \\ e^{2i(\phi_g+\alpha_g)}e^{i(\phi+\alpha+\alpha_a-\alpha_b)} &= -1. \end{aligned} \tag{52}$$

Solving these equations gives:

$$\begin{aligned} \phi+\alpha+\alpha_a-\alpha_b &= \pi \\ \phi_g+\alpha_g &= 0. \end{aligned} \tag{53}$$

Therefore, we set $\phi_g = -\alpha_g$, $\alpha_a = \pi - \phi - \alpha$, and $\alpha_b = 0$ to ensure that $ab^\dagger$ is similar to X. Now, all the parameters in a and b are specified, and the procedure from the previous Section can be used to compute the gate sequence implementing the desired transformation. This concludes the description of each individual step in our LNNs implementation of optimal, exact three-qubit state preparation.

C.5. CX gate count for UCG Exact Method on LNN Architecture

Bergholm *et al* [17] showed that exact QSP on n qubits can be reduced to a sequence of n uniformly controlled single-qubit gates, where the kth gate uses qubits $[0\ldots n-1-k]$ as the controls, and qubit $n-k$ as the target. The last UCG is a single-qubit rotation on qubit 0. Next, the authors showed that, on LNN architectures, a uniformly controlled single-qubit gate using controls $[0\ldots c-1]$ and target qubit c can be implemented, up to a diagonal gate, as

$$\mathrm{UCG}(c) = \prod_{i=0}^{2^c-1} \mathrm{CX\ chain}(c-1-z(i), c)\, U(i), \tag{54}$$

for some choice of single-qubit rotations $U(i)$, where $z(i)$ is the number of trailing zeroes in the binary representation of i (define $z(0) = c-1$) and CX chain(i,j) is a sequence of CX gates:

$$\begin{aligned} CX\,\mathrm{chain}(i,j) = {} & CX(i,i+1)\,CX(i+1,i+2)\ldots CX(j-2,j-1)\,CX(j-1,j) \\ & \times CX(j-2,j-1)\ldots CX(i+1,i+2)\,CX(i,i+1). \end{aligned} \tag{55}$$

The number of CX gates in CX chain(i,j) is $2(j-i)-1$. Then, the number of CX gates needed to implement UCG(c) is

$$\begin{aligned} CX\,\mathrm{count}(\mathrm{UCG}(c)) &= \sum_{i=0}^{2^c-1}(2(1+z(i))-1) \\ &= \sum_{i=0}^{2^c-1}(2z(i)+1) \\ &= 2^c + 2\sum_{i=0}^{2^c-1} z(i). \end{aligned} \tag{56}$$

Since i takes on all non-negative integers with c bits inthe sum, there are 2^{c-m-1} instances where $z(i) = m$, for $0 \leqslant m \leqslant c-2$. However, $m = c-1$ will show up twice because $z(0)$ is defined to be $c-1$. Then,

$$\begin{aligned} \sum_{i=0}^{2^c-1} z(i) &= \sum_{m=0}^{c-2} m 2^{c-m-1} + 2(c-1) = 2^c - 2 \\ CX\,\mathrm{count}(\mathrm{UCG}(c)) &= 2^c + 2(2^c-2) = 3\times 2^c - 4. \end{aligned} \tag{57}$$

This would be the number of CX gates required to implement UCG(c) on an LNN architecture, however, [17] introduces an optimization: by swapping qubits c and $c-1$, all of the CX chain$(c-1-z(i), c)$ terms in equation (54), where $c-1-z(i) < c-1$, become CX chain$(c-1-z(i), c-1)$. This saves 2 CX gates for each of those terms. In addition, for the CX chains where $c-1-z(i) = c-1$, the CX chain$(c-1-z(i), c)$ term becomes CX chain$(c-1, c)$, which is the same thing. The swapping procedure costs 6 CX gates (3 to swap the qubits, 3 to swap back at the end), but it saves 2 CX gates for each term where $c-1-z(i) < c-1 \implies z(i) > 0$. There are 2^{c-1} such terms, for a net savings of $2^c - 6$ CX gates. Thus,

$$CX\,\mathrm{count,\ optimized}(\mathrm{UCG}(c)) = 3\times 2^c - 4 - (2^c - 6) = 2\times 2^c + 2. \tag{58}$$

Table 3. Average fidelity, *CX* gate count, and computational effort required to prepare protein-encoded quantum states with matrix product states (MPS) approach with different values of bond dimension hyperparameter.

Bond dimension	Avg. Fidelity	Avg. CX gate count	Classical runtime
2	0.00045	29.8	1.89
4	0.0016	181.8	12.86
8	0.0006	963.8	528.03
16	0.0024	4353.8	4393.3

We introduce one additional optimization: the UCG(2), UCG(1), and UCG(0) gates at the end can be replaced by exact, optimal three-qubit state preparation. Then, we can compute the total number of *CX* gates required to prepare an n-qubit quantum state ($n > 3$) using the UCG method:

$$CX \text{ count, UCG on} n\text{qubits} = 3 + \sum_{k=3}^{n-1} \left(2 \times 2^k + 2\right) = 3 + 2\left(2^n - 8\right) + 2\left(n - 3\right) = 2 \times 2^n + 2n - 19. \quad (59)$$

Appendix D. Benchmarking MPSs

We conducted additional experiments using the MPSs approach for state preparation for 10-qubit QSP tasks using protein-encoded quantum states, evaluating average fidelity, *CX* gate count, and classical computational runtime across different bond dimensions. The computational results presented in the table below indicate that the fidelity of the quantum state prepared with MPS is dependent on the bond dimension hyperparamter. There is also a clear exponential increase in both the *CX* gate count and classical runtime as the bond dimension increases. The *CX* gate count grows from approximately 30 gates at bond dimension 2 to over 4300 gates at bond dimension 16, while the classical runtime escalates dramatically from 1.89 units to 4393.3 units. This suggests that higher bond dimensions, while potentially offering marginally better fidelity in some cases, come at a significant cost in terms of circuit complexity and computational resources. MPS is suited for encoding smooth functions like probability distributions as they typically correspond to low-entanglement states [62, 63]. As the classical data to be encoded becomes more complex, as can be case with chemical data, the data-encoded quantum states become highly entangled. In these cases, MPS struggles to efficiently represent such states [64, 65].

ORCID iD

Fengqi You https://orcid.org/0000-0001-9609-4299

References

[1] Shor P W 1994 Algorithms for quantum computation: discrete logarithms and factoring *Proc. 35th Annual Symp. on Foundations of Computer Science* (IEEE) pp 124–34
[2] Lloyd S 1996 Universal quantum simulators *Science* **273** 1073–8
[3] Berry D W, Childs A M, Cleve R, Kothari R and Somma R D 2015 Simulating Hamiltonian dynamics with a truncated Taylor series *Phys. Rev. Lett.* **114** 090502
[4] Berry D W, Childs A M and Kothari R 2015 Hamiltonian simulation with nearly optimal dependence on all parameters *2015 IEEE 56th Annual Symp. on Foundations of Computer Science* (IEEE) pp 792–809
[5] Low G H and Chuang I L 2017 Optimal Hamiltonian simulation by quantum signal processing *Phys. Rev. Lett.* **118** 010501
[6] Low G H and Chuang I L 2019 Hamiltonian simulation by qubitization *Quantum* **3** 163
[7] Harrow A W, Hassidim A and Lloyd S 2009 Quantum algorithm for linear systems of equations *Phys. Rev. Lett.* **103** 150502
[8] Wossnig L, Zhao Z and Prakash A 2018 Quantum linear system algorithm for dense matrices *Phys. Rev. Lett.* **120** 050502
[9] Outeiral C, Strahm M, Shi J, Morris G M, Benjamin S C and Deane C M 2021 The prospects of quantum computing in computational molecular biology *Wiley Interdiscip. Rev.-Comput. Mol. Sci.* **11** e1481
[10] Batra K, Zorn K M, Foil D H, Minerali E, Gawriljuk V O, Lane T R and Ekins S 2021 Quantum machine learning algorithms for drug discovery applications *J. Chem. Inf. Model.* **61** 2641–7
[11] Ajagekar A and You F 2022 New frontiers of quantum computing in chemical engineering *Korean J. Chem. Eng.* **39** 811–20
[12] Andersson M P, Jones M N, Mikkelsen K V, You F and Mansouri S S 2022 Quantum computing for chemical and biomolecular product design *Curr. Opin. Chem. Eng.* **36** 100754
[13] Doga H, Raubenolt B, Cumbo F, Joshi J, DiFilippo F P, Qin J, Blankenberg D and Shehab O 2024 A perspective on protein structure prediction using quantum computers *J. Chem. Theory Comput.* **20** 3359–78
[14] Aaronson S 2015 Read the fine print *Nat. Phys.* **11** 291–3
[15] Pal S, Bhattacharya M, Lee S-S and Chakraborty C 2024 Quantum computing in the next-generation computational biology landscape: from protein folding to molecular dynamics *Mol. Biotechnol.* **66** 163–78
[16] Shende V V, Bullock S S and Markov I L 2006 Synthesis of quantum-logic circuits *IEEE Trans. Comput.-Aided Des. Integr.* **25** 1000–10

[17] Bergholm V, Vartiainen J J, Möttönen M and Salomaa M M 2005 Quantum circuits with uniformly controlled one-qubit gates *Phys. Rev.* A **71** 052330
[18] Plesch M and Brukner Č 2011 Quantum-state preparation with universal gate decompositions *Phys. Rev.* A **83** 032302
[19] Sun X, Tian G, Yang S, Yuan P and Zhang S 2023 Asymptotically optimal circuit depth for quantum state preparation and general unitary synthesis *IEEE Trans. Comput.-Aided Des. Integr.* pp 1–1
[20] Zhang X-M, Li T and Yuan X 2022 Quantum state preparation with optimal circuit depth: implementations and applications *Phys. Rev. Lett.* **129** 230504
[21] Gleinig N and Hoefler T 2021 An efficient algorithm for sparse quantum state preparation *2021 58th ACM/IEEE Design Automation Conf. (DAC)* (IEEE Press) pp 433–8
[22] Malvetti E, Iten R and Colbeck R 2021 Quantum circuits for sparse isometries *Quantum* **5** 412
[23] Nakaji K, Uno S, Suzuki Y, Raymond R, Onodera T, Tanaka T, Tezuka H, Mitsuda N and Yamamoto N 2022 Approximate amplitude encoding in shallow parameterized quantum circuits and its application to financial market indicators *Phys. Rev. Res.* **4** 023136
[24] Mitarai K, Negoro M, Kitagawa M and Fujii K 2018 Quantum circuit learning *Phys. Rev.* A **98** 032309
[25] Rivera-Dean J, Huembeli P, Acín A and Bowles J 2021 Avoiding local minima in variational quantum algorithms with neural networks (arXiv:2104.02955)
[26] Melnikov A A, Termanova A A, Dolgov S V, Neukart F and Perelshtein M 2023 Quantum state preparation using tensor networks *Quantum Sci. Technol.* **8** 035027
[27] Iaconis J, Johri S and Zhu E Y 2024 Quantum state preparation of normal distributions using matrix product states *npj Quantum Inf.* **10** 15
[28] Holmes A and Matsuura A Y 2020 Efficient quantum circuits for accurate state preparation of smooth, differentiable functions *2020 IEEE Int. Conf. on Quantum Computing and Engineering (QCE)* (IEEE) pp 169–79
[29] Bernal D E, Ajagekar A, Harwood S M, Stober S T, Trenev D and You F 2022 Perspectives of quantum computing for chemical engineering *AIChE J.* **68** e17651
[30] Gujju Y, Matsuo A and Raymond R 2024 Quantum machine learning on near-term quantum devices: current state of supervised and unsupervised techniques for real-world applications *Phys. Rev. Appl.* **21** 067001
[31] Preskill J 2018 Quantum computing in the NISQ era and beyond *Quantum* **2** 79
[32] Xu Y, Verma D, Sheridan R P, Liaw A, Ma J, Marshall N M, McIntosh J, Sherer E C, Svetnik V and Johnston J M 2020 Deep dive into machine learning models for protein engineering *J. Chem. Inf. Model.* **60** 2773–90
[33] Bravyi S, Dial O, Gambetta J M, Gil D and Nazario Z 2022 The future of quantum computing with superconducting qubits *J. Appl. Phys.* **132** 160902
[34] Saeedi M, Wille R and Drechsler R 2010 Synthesis of quantum circuits for linear nearest neighbor architectures *Quantum Inf. Process.* **10** 355–77
[35] Bepler T and Berger B 2019 Learning protein sequence embeddings using information from structure (arXiv:1902.08661)
[36] Yang K K, Wu Z, Bedbrook C N, Arnold F H and Wren J 2018 Learned protein embeddings for machine learning *Bioinformatics* **34** 2642–8
[37] Zhou Z, Ji Y, Li W, Dutta P, Davuluri R and Liu H 2023 Dnabert-2: efficient foundation model and benchmark for multi-species genome (arXiv:2306.15006)
[38] Dalla-Torre H *et al* 2025 Nucleotide transformer: building and evaluating robust foundation models for human genomics *Nat. Methods* **22** 287–97
[39] Nguyen E *et al* 2024 Hyenadna: long-range genomic sequence modeling at single nucleotide resolution *Advances in Neural Information Processing Systems* vol 43177–201
[40] Jaeger S, Fulle S and Turk S 2018 Mol2vec: unsupervised machine learning approach with chemical intuition *J. Chem. Inf. Model.* **58** 27–35
[41] Mozafari F, De Micheli G and Yang Y 2022 Efficient deterministic preparation of quantum states using decision diagrams *Phys. Rev.* A **106** 022617
[42] Ben-Or M and Hassidim A 2005 Fast quantum byzantine agreement *Proc. 37th Annual ACM Symp. on Theory of Computing (STOC'05)* (Association for Computing Machinery) pp 481–5
[43] Sierra-Sosa D, Telahun M and Elmaghraby A 2020 Tensorflow quantum: impacts of quantum state preparation on quantum machine learning performance *IEEE Access* **8** 215246–55
[44] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 Quantum machine learning *Nature* **549** 195–202
[45] Consortium U 2015 Uniprot: a hub for protein information *Nucleic Acids Res.* **43** D204–12
[46] Maldonado T J, Flick J, Krastanov S and Galda A 2022 Error rate reduction of single-qubit gates via noise-aware decomposition into native gates *Sci. Rep.* **12** 6379
[47] Žnidarič M, Giraud O and Georgeot B 2008 Optimal number of controlled-not gates to generate a three-qubit state *Phys. Rev.* A **77** 032320
[48] Zhang K, Hsieh M-H, Liu L and Tao D 2020 Toward trainability of quantum neural networks (arXiv:2011.06258)
[49] Grimsley H R, Economou S E, Barnes E and Mayhall N J 2019 An adaptive variational algorithm for exact molecular simulations on a quantum computer *Nat. Commun.* **10** 3007
[50] Elnaggar A *et al* 2020 Prottrans: towards cracking the language of life's code through self-supervised learning *IEEE Trans. Pattern Anal. Mach. Intell.* **44** 7112–27
[51] Temme K, Bravyi S and Gambetta J M 2017 Error mitigation for short-depth quantum circuits *Phys. Rev. Lett.* **119** 180509
[52] Li Y and Benjamin S C 2017 Efficient variational quantum simulator incorporating active error minimization *Phys. Rev.* X **7** 021050
[53] Endo S, Benjamin S C and Li Y 2018 Practical quantum error mitigation for near-future applications *Phys. Rev.* X **8** 031027
[54] Rattew A G, Sun Y, Minssen P and Pistoia M 2021 The efficient preparation of normal distributions in quantum registers *Quantum* **5** 609
[55] Paetznick A and Svore K M 2013 Repeat-until-success: non-deterministic decomposition of single-qubit unitaries (arXiv:1311.1074)
[56] Gottesman D 2013 Fault-tolerant quantum computation with constant overhead (arXiv:1310.2984)
[57] Bravyi S B and Kitaev A Y 1998 Quantum codes on a lattice with boundary (arXiv:quant-ph/9811052)
[58] Fowler A G, Mariantoni M, Martinis J M and Cleland A N 2012 Surface codes: towards practical large-scale quantum computation *Phys. Rev.* A **86** 032324
[59] Kitaev A Y 1997 Quantum computations: algorithms and error correction *Russ. Math. Surv.* **52** 1191
[60] Dawson C M and Nielsen M A 2005 The solovay-kitaev algorithm (arXiv:quant-ph/0505030)

[61] Bravyi S and Kitaev A 2005 universal quantum computation with ideal clifford gates and noisy ancillas *Phys. Rev.* A **71** 022316
[62] García-Ripoll J J 2021 Quantum-inspired algorithms for multivariate analysis: from interpolation to partial differential equations *Quantum* **5** 431
[63] Vidal G 2003 Efficient classical simulation of slightly entangled quantum computations *Phys. Rev. Lett.* **91** 147902
[64] Lubasch M, Joo J, Moinier P, Kiffner M and Jaksch D 2020 Variational quantum algorithms for nonlinear problems *Phys. Rev.* A **101** 010301
[65] Eisert J, Cramer M and Plenio M B 2008 Area laws for the entanglement entropy-a review (arXiv:0808.3773)