Resource analysis of quantum algorithms for coarse-grained protein folding models

Hanna Linn,* Isak Brundin, Laura García-Álvarez, and Göran Johansson
Department of Microtechnology and Nanoscience (MC2), Chalmers University of Technology, SE-412 96 Göteborg, Sweden

(Received 19 December 2023; accepted 24 May 2024; published 26 July 2024)

Protein folding processes are a vital aspect of molecular biology that is hard to simulate with conventional computers. Quantum algorithms have been proven superior for certain problems and may help tackle this complex life science challenge. We analyze the resource requirements for simulating simplified yet computationally challenging protein folding models on a quantum computer, assessing the feasibility of these existing approaches in the current and near-future technological landscape. We calculate the minimum number of qubits, interactions, and two-qubit gates necessary to build a heuristic quantum algorithm with the specific information of a folding problem. Particularly, we focus on the resources needed to build quantum operations based on the Hamiltonian linked to the protein folding models for a given amino acid count. Such operations are a fundamental component of these quantum algorithms, guiding the evolution of the quantum state for efficient computations. Specifically, we study coarse-grained folding models on the lattice and the fixed backbone side-chain conformation model and assess their compatibility with the constraints of existing quantum hardware given different bit encodings. We conclude that the number of qubits required falls within current technological capabilities. However, the limiting factor is the high number of interactions in the Hamiltonian, resulting in a quantum gate count unavailable today.

DOI: 10.1103/PhysRevResearch.6.033112

I. INTRODUCTION

Shor's and Grover's algorithms demonstrate exponential and polynomial speedups compared to classical methods for the practical problems of prime factorization [1] and unsorted searching [2]. This breakthrough ignited the pursuit of further quantum algorithms and use cases, aiming to achieve quantum speedup in computations for real-world applications. However, reaching such an advantage typically requires fully error-corrected devices yet to be available, with around 10^5 qubits [3]. Until such platforms become a reality, the community explores algorithms suitable for the current noisy intermediate-scale quantum (NISQ) computers [4], with 50 to a few hundred noisy qubits. The current efforts to realize a quantum computer involve various approaches, such as superconducting qubits [5], trapped ion qubits [6], and photonic qubits [7], each with its technical challenges and tradeoffs regarding qubit count, connectivity, and coherence time. The latter limits the gate fidelities so that the computation will be dominated by noise, usually well before performing even 1000 gates on each qubit.

In the NISQ era, hybrid quantum-classical algorithms have attracted considerable attention for their ability to harness existing hardware capabilities and potentially provide a quantum advantage for specific computational problems [8], with one notable example being the quantum approximate optimization algorithm (QAOA) [9]. A customized variational ansatz containing solutions to a problem depends on a finite number of classically optimizable parameters related to a heuristic quantum algorithm. This ansatz construction commonly involves using problem-specific information, such as the energy or cost function of a given optimization problem. Hybrid algorithms are highly adaptable and thereby find applications in various domains, including chemistry [10], machine learning [11], and optimization [9], as well as applications in the field of protein folding [12,13].

Studying protein folding is essential for understanding how proteins gain their functional three-dimensional structures, comprehending the protein's biological functions, and designing effective therapeutic interventions. The folding from an amino acid sequence to a stable structure is deeply intricate [14,15], and simulating the dynamics represents a complex optimization problem where one wants to find the conformation with the lowest score according to an energy function [16,17]. The field of in silico protein structure prediction has seen massive success recently with classical heuristics [18–20]. Before the availability of computational resources enabling these fully atomistic predictions, coarse-graining models served as an intermediary step and continue to be essential in multiscale modeling today. Coarse-grained models decrease the sampling space and lower the complexity of the problem by assuming various levels of reduced polypeptide chain representation [21]. As the problem remains challenging for classical deterministic algorithms, Monte Carlo or other heuristics have become the field standard for coarse-grained models, performing well within ranges of a few tens of amino acids [22,23]. Similarly, due to the limited quantum resources currently available, quantum algorithms must

*Contact author: hannlinn@chalmers.se

consider these simplifications. We focus on the coarse-grained models previously explored in the field of quantum algorithms to give an overview of the field: the lattice model [24,25] with hydrophobic-polar (HP) energies [26] or Miyazawa-Jernigan (MJ) energies [27], and an off-lattice side-chain conformation-based model with a fixed backbone [28,29]. The HP-lattice model is used to study protein thermodynamics, folding dynamics, and evolution [30], and the side-chain conformation-based model with a fixed backbone plays a role within broader algorithms for more complex models [31].

The first proof-of-concept proposals for finding optimal protein folds with quantum devices consider only small peptides due to the quantum hardware limitations. Even if these simplified models for peptide sampling are not used for folding predictions in realistic contexts, they remain computationally demanding and can provide qualitatively relevant results. Therefore, the quantum computing community considers them suitable testbeds for quantum heuristic algorithms. Early formulations for quantum annealers include the HP-lattice model on a square grid, with the amino acid coordinates encoded with binary strings [32]. The high resource demand of this initial approach motivated the search for new problem formulations, leading to a divide-and-conquer method and a turn-based encoding of the protein conformation. Consequently, a quantum annealer successfully found the optimal fold of a chain with six amino acids for a model with MJ energies [33], and this formulation was adapted to QAOA for an ion-trap experiment [34]. Subsequent works lowered the quantum operations needed [35], which prompted further experiments on a quantum annealer tackling the folding of a ten-amino acid chain on a planar lattice and an eight-amino acid chain on a cubic lattice [36]. The quantum algorithm improvements for constrained optimization problems resulted in the use of the quantum alternating operator ansatz [37] to fold a four-amino acid long chain on an ion-trap quantum computer and further explorations to incorporate efficiently the problem constraints in the algorithm formulation [12]. Such strategic engineering of the problem formulation can also improve the success of the folding problem on quantum annealers [38]. Moreover, the field benefited from a new resource-efficient model with MJ energies and a tetrahedral lattice [13]. This model was first implemented experimentally on a superconducting circuit quantum computer for a seven-amino acid long chain [13]. Despite the fact that numerical simulations of QAOA for this tetrahedral lattice model show lower performance, even for a sequence of four amino acids [39], when combining this model with other algorithms such as the digitized-counterdiabatic quantum algorithm, it succeeded in folding a nine-amino acid chain on two superconducting circuit gate-based quantum devices and an ion-trap platform [40]. Likewise, parallel efforts led to the realization of an HP-lattice model for a 14-amino acid long chain on a quantum annealer. The model reduced the qubits needed for a coordinate-based conformation encoding by including lattice symmetries [41]. In these early stages, the simplicity of the HP-lattice model compared to higher-resolution descriptions positions it among the central models for different quantum algorithms, including Grover-based protocols [42]. Recently, the field has also expanded into off-lattice models. These works range from formulations of peptide packing for a quantum annealer using side-chain conformation-based models [43,44] to methods using deep learning for initial state generation followed by a quantum Metropolis-Hastings algorithm to decide parametrized torsion angles of a tetrapeptide [45].

In every previous formulation, the authors have made different choices regarding the protein folding model and its encoding into the quantum computer. The models' resolution and translation to quantum variables impact the algorithm's time and space complexity. This translation includes representing a specific folding or conformation with qubits and expressing the model's energy function using interactions between them. In essence, the resulting problem formulation as a quantum spin model includes the interactions between the coarse-grained beads of the protein and the folding constraints in the so-called cost Hamiltonian. We can customize this Hamiltonian to focus on specific aspects of the problem or exploit particular symmetries or properties. The size and complexity of the cost Hamiltonian will affect the corresponding required quantum circuit. That is, it impacts the run time or circuit depth, the number of qubits, and how they interact, which will, in turn, influence the choice of hardware. Optimal decisions in the encoding step can reduce the need for quantum hardware resources, while suboptimal ones may strain connectivities and operations allowed in the devices. Due to the diverse capabilities inherent in quantum platforms [46], resource tradeoffs arise contingent upon the chosen encoding strategy.

Improving quantum algorithms requires analyzing how the problem encoding can affect the necessary resources and the demands on current quantum technology. Accordingly, the community has studied encoding resource tradeoffs for quantum and discrete optimization problems [47,48]. A more generalized approach includes an extensive library of hardware-independent formulations of these problems for quantum computing given different encodings [49]. In contrast, another analysis focused on the feasibility of several hybrid quantum-classical algorithms on current quantum computers for maximum independent set—an optimization problem over binary variables. There, as the encodings do not play a role, they detailed the quantum gates and classical resources needed, showcasing tradeoffs between gate decomposition methods for different quantum hardware [50].

In this paper, we explore several encodings of discrete optimization formulations of protein folding into quantum variables and analyze the resource requirements given the characteristics of different quantum hardware. Our objective is to assess the suitability of gate-based quantum computers, specifically NISQ devices, to potentially fold proteins of realistic size. Human proteins have a median length of 375 amino acids [51], whereas clinical target proteins tend to be about 414 amino acids long on average [52]. So, what kind of quantum computer would we need to fold 100 amino acids? Since fully atomistic descriptions used for folding predictions are beyond the capabilities of current quantum devices, we examine the scaling of lattice protein folding and side-chain packing models, earlier addressed by the quantum computing community. We detail the required quantum resources—the number of qubits, interactions, and the resulting two-qubit

gates—linked to the unary, binary, and block-unary binary (BU-binary) encodings for these computationally challenging models. In particular, we only estimate the resources needed to create the building block operation linked to the cost Hamiltonian in the quantum algorithm. That is, we provide a lower bound of the resources required to address these protein folding models with quantum heuristics for current technology by presenting the resources for a single cost layer of the QAOA. It remains uncertain whether these quantum algorithms can succeed, and additionally, how many repetitions of the building block the algorithm requires to perform well is yet to be determined.

The paper is structured as follows. In Sec. II, we describe coarse-grained protein models and, in particular, the discrete variable formulation of HP-lattice models and side-chain conformation-based models. In Sec. III, we present the unary, binary, and BU-binary encodings for translating discrete variables into qubits and their application to the previous models. Further, in Sec. IV, we analyze the tradeoffs associated with qubit growth, gates, and the size of the unfeasible solution set in the encoding Hamiltonians. Finally, Sec. V concludes the discussion by summarizing the key findings, highlighting their implications, and suggesting future research directions.

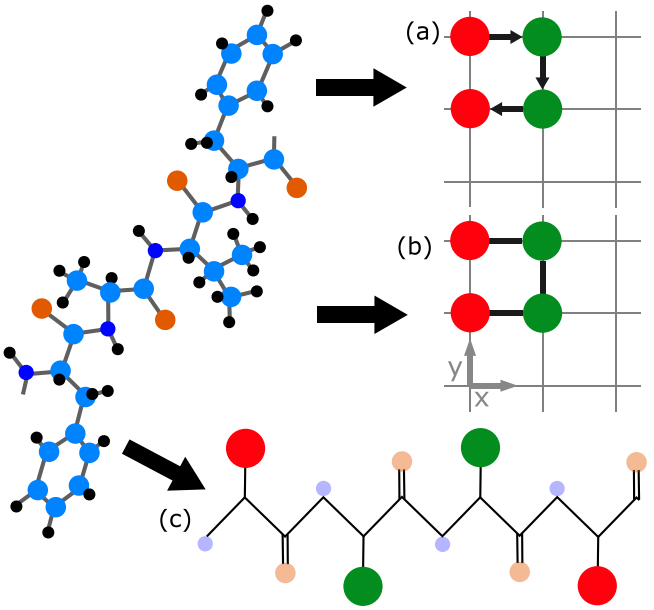

FIG. 1. Fully atomistic depiction of a four amino acid chain and the corresponding coarse-grain model representations for the (a) turn-based lattice model, (b) coordinate-based lattice model, and (c) side-chain conformation-based model.

II. COARSE-GRAINED PROTEIN MODELS

Studying proteins is challenging due to the size of the systems and their interactions. We focus on coarse-grained protein models that lower the degrees of freedom in the polypeptide representation [21]. Different simplifications lead to computationally advantageous low-resolution models that capture different system properties and behaviors, enabling the study of protein folding mechanisms or protein structure prediction, among others. These low-resolution models characterize larger systems over longer timescales and may be combined with atomistic simulations in multiscale modeling schemes.

We consider coarse-grained models designed to understand the protein folding process and suitable for quantum computing architectures. The models include two principal reductions to ensure their feasibility with the state-of-the-art and near-term quantum technology. First, chains of beads with different properties represent groups of atoms in amino acids. Second, we use discrete representations of the structures' geometry—the chain of beads is placed on lattice grids or discretized spatial orientations—such that the finding of the optimal fold can be formulated as a discrete optimization problem.

The chain representation of the protein varies according to the level of resolution. Despite the huge simplifications of these lattice models, finding their optimal protein conformations is an NP-hard problem that challenges the capabilities of conventional computers [53–56]. Moreover, other methods replace only partially the amino acid with a bead, such as ROSETTA's centroid representation, where the backbone remains atomistic, and the beads replace the side chains. We consider these approximations and focus on coarse-grained models addressed previously by the quantum computing community, which can be formulated as discrete optimization problems, namely, lattice models [12,13,32–36,38,41] and the side-chain conformation-based model [43,44]. For the sake of simplicity, we consider two-dimensional square and three-dimensional cubic lattices.

To compare the resource requirements—the number of qubits, interactions, and gates—for the quantum hardware implementation of the different models, we use a general formulation that accounts for the number of possible conformations of each model. Given a sequence of N amino acids $(a_1, \ldots, a_N)$ for which each amino acid a_i can take a conformation from the finite set of integers, $R_i = \{1, 2, \ldots, c_i\}$, we define a vector with the cardinalities of the sets of conformations $c_i = n(R_i)$ as

$$\mathbf{C} = (c_1, c_2, \ldots, c_N). \tag{1}$$

Given a choice $r_i \in R_i$ for each amino acid, we create a bitstring $x = r_1 r_2 \ldots r_N$ encoding a protein conformation by concatenating the binary representation of each integer r_i. Various methods exist to convert the integer information into binary strings, presented in Sec. III. To perform the discrete optimization, we need an energy function that assesses the energy of the bitstring as we want to find the bitstring encoding the protein conformation with the lowest energy. This function combines negative potentials from interacting amino acids with positive penalty terms, maintaining physical constraints.

We use this framework to describe the lattice and side-chain conformation-based models in Secs. II A and II B, respectively. See Fig. 1 for a schematic representation of the coarse-grained simplifications. The vector $\mathbf{C}$ varies across models. In lattice models, it is insufficient to describe allowed bitstrings entirely. In coordinate-based models, extra frameworks are needed for physical conformation, while turn-based models require additional qubits to assess bead interactions.

A. HP-lattice models

In the HP-lattice model, a protein is represented by a chain $(a_1, \ldots, a_N)$ of hydrophobic and polar beads, $a_i \in \{\mathrm{H,P}\}$, placed on a lattice. This model consists of an energy function whose ground state corresponds to the optimal fold that minimizes the exposure of hydrophobic residues to the solvent while penalizing nonphysical configurations. First, one considers a pairwise potential that depends on the distance between beads and accounts for the number of nearest-neighbor hydrophobic beads in the conformation. Second, we impose constraints on the possible folds such that the chain is not overlapping—with two amino acids occupying the same lattice site—or broken. With lattice grids, the distance in the energy function can be formulated in terms of integer variables, leading to a classical discrete optimization problem.

We study the two most common ways of encoding the sequence placement on the lattice and, therefore, the distance between beads. On the one hand, seminal works in quantum computation use coordinate-based encodings, for which one needed at least $DN \log_2 N$ qubits, with D the lattice dimension and N the number of amino acids [32]. On the other hand, further developments introduced a turn-based encoding, in which the eventual translation to quantum hardware reduced the qubit requirements at the cost of more operations [33].

1. Coordinate-based HP-lattice model

In this model, the protein conformation is encoded with the lattice coordinates of each bead. That is, each integer component of the vector in Eq. (1) is the number of possible positions of each chain component a_i on the lattice grid. This paper focuses on the model presented in Ref. [41]. We consider the additional checkerboard symmetry that divides the number of available positions for each bead in half [41]. Essentially, we label the grid locations and chain beads as odd or even, such that each bead can only be placed in a location with the same parity. In the two-dimensional case, we repeat a square unit cell in two directions to generate a lattice of side lengths L_1 and L_2. Therefore, the number of possible available conformations for each bead is given by the cardinality vector:

$$\mathbf{C}^{\mathrm{coord}}_{\mathrm{HP\text{-}square}} = \left(\left\lceil \frac{L_1 L_2}{2} \right\rceil, \left\lfloor \frac{L_1 L_2}{2} \right\rfloor, \left\lceil \frac{L_1 L_2}{2} \right\rceil, \ldots \right). \quad (2)$$

We also consider a three-dimensional lattice with side lengths of L_1, L_2, and L_3 created from a cubic unit cell. In this case, the number of possible locations for the beads can take values up to

$$\mathbf{C}^{\mathrm{coord}}_{\mathrm{HP\text{-}cubic}} = \left(\left\lceil \frac{L_1 L_2 L_3}{2} \right\rceil, \left\lfloor \frac{L_1 L_2 L_3}{2} \right\rfloor, \left\lceil \frac{L_1 L_2 L_3}{2} \right\rceil, \ldots \right). \quad (3)$$

The energy function also utilizes checkerboard symmetry, evaluating the pairwise potential for adjacent even and odd sites. Penalty terms are used to enforce three constraints that help to eliminate nonphysical bitstrings. The first constraint ensures that each bead is assigned to exactly one lattice site by penalizing more than one conformation per bead. The second constraint involves calculating a self-avoidance term that counts the number of beads that have chosen the same conformation, thus preventing two amino acids from occupying the same lattice point. The third constraint maintains sequence order by checking the distance between consecutive beads on the string [41].

2. Turn-based HP-lattice model

We describe the protein conformation by tracking the directions in which each amino acid turns when placed sequentially on the lattice. The distance between beads and interactions in the HP-lattice energy function can be rewritten accordingly [13,35,36].

This turn-based encoding of the beads' placements reduces the possible values of the location variables at the cost of increasing the number of interactions. The elements of the cardinality vector defined in Eq. (1) correspond here to the lattice coordination number, and thus

$$\mathbf{C}^{\mathrm{turn}}_{\mathrm{HP\text{-}square}} = (4, \ldots, 4), \quad (4)$$

$$\mathbf{C}^{\mathrm{turn}}_{\mathrm{HP\text{-}cubic}} = (6, \ldots, 6). \quad (5)$$

The simplicity of encoding the conformation for each bead in the sequence makes for a complex energy function that calls for extra resources, i.e., additional qubits. Euclidean distances between beads are computed by summing the turns in all directions and converting them into coordinates based on the protein's conformation. Auxiliary qubits are employed to store these distances, enabling the calculation of pairwise interactions between sequence beads. The auxiliary bits can also be used to construct a penalty term that discourages solutions with zero distance between beads, that is, overlapping beads.

Two versions of the turn-based model are present in the literature: the turn ancilla encoding, which places the interaction information in auxiliary qubits, and the turn circuit encoding, which uses multiqubit terms to keep track of this information. The first one leads to a quadratic qubit growth with the number of amino acids. The latter uses nonunitary half adders and XNOR gates [12,35], which may be a more resource-efficient encoding. We have not considered this encoding in our analysis, but note that many half adders may add auxiliary bits to be implemented to the otherwise linear qubit requirement [57].

B. Side-chain conformation-based models

Several protein folding algorithms alternate between side-chain packing and backbone optimization to predict the optimal protein fold. The side-chain conformation-based models are used to find the optimal side-chain packing given a fixed backbone sequence and variable side-chain conformations. In contrast to the HP-lattice model, these models can reach a higher resolution for this intermediate step as we carry more information on the amino acid composition. We focus on the side-packing problem, which is NP-complete [58]. We consider a discrete number of torsional angles or rotamers describing a given backbone's amino acid side-chain conformations. Therefore, following Eq. (1), for each amino acid a_i on a sequence, its side chain can take a limited number of rotational conformations c_i, with $\prod_i c_i$ the number of all feasible protein structures.

In these rotamer-based models, a function determines the energy of each protein conformation and guides the optimization toward the most energetically favorable structures. A commonly used function is the ROSETTA energy function, which combines physics-based potentials with knowledge-based energies to return an energy score for a given structure [59]. This energy function is a fundamental component of the ROSETTA software suite used for protein structure prediction and design, already considered for optimization with quantum annealers [43].

Without loss of generality, we use an energy-scoring function that considers all pairwise interactions between rotamers at different amino acids. That is, given the conformations $r_i \in R_i$ from the set of possible conformations R_i of the side chain associated with each amino acid a_i, the energy function is given by

$$E(\mathbf{r}) = \sum_{i=2}^{N} \sum_{j<i}^{N-1} E_{ij}(r_i, r_j), \tag{6}$$

with $\mathbf{r} = (r_1, r_2, \ldots, r_N)$ a vector of integers associated with a particular rotamer selection and protein structure. The energy $E(r_i, r_j)$ depends on the pair of rotamers at position i and j and contains the two-body energy between the pair.

III. ENCODINGS OF DISCRETE VARIABLES INTO QUBITS

A function scoring the energy of a protein conformation $x = r_1 r_2 \ldots r_N$ can be translated to a quantum cost Hamiltonian. With that aim, we consider the concatenation of every integer r_i represented with binary variables, such that $x = x_1 x_2 \ldots x_M$, with $x_i \in \{0, 1\}$. Then, each classical variable x_i maps to a quantum variable σ_i^z that can take values $\{1, -1\}$ as $x_i \to (1 - \sigma_i^z)/2$. This cost Hamiltonian, used in quantum annealing [60] and in gate-based quantum algorithms [9], can be represented as a k-local spin Hamiltonian:

$$H_{\text{cost}} = \sum_i h_i \sigma_i^z + \sum_{ij} J_{ij} \sigma_i^z \sigma_j^z + \sum_{ijk} J_{ijk} \sigma_i^z \sigma_j^z \sigma_k^z + \ldots, \tag{7}$$

with interaction terms involving at most k qubits. Here, h_i, J_{ij}, and J_{ijk} correspond to the single-qubit, two-qubit, and many-qubit energy coefficients, respectively. The Pauli z matrix acting on the jth qubit is σ_j^z. Analyzing the necessary resources involves focusing on this cost Hamiltonian, which encodes the problem and guides the system from the initial to the optimal final state.

A. Encodings: Unary, binary, and BU-binary

As mentioned earlier, the protein conformation is encoded into a bitstring $x = r_1 r_2 \ldots r_N$. This encoding involves selecting one conformation $r_i \in R_i$ for each amino acid. Different encodings of the integers r_i result in varying resource usage.

1. Unary

In the unary encoding, an integer is denoted by a solitary one in the binary representation, positioned at the bit corresponding to the integer's value. Each set of conformation for a given amino acid a_i is encoded by a substring of length c_i, where a single 1 represents one conformation, and the rest of the substring is zero (see Table I). Each substring of length c_i has the Hamming weight one to ensure only one conformation is chosen. The total length of the bitstring is $\sum_i c_i$ (see Table II), and the total Hamming weight is equal to the number of amino acids M.

TABLE I. The integer to unary, binary, and BU-binary encodings with block size variable $g = 3$.

Decimal	Unary	Binary	BU-binary$_{g=3}$
0	10000	000	00 01
1	01000	001	00 10
2	00100	010	00 11
3	00010	011	01 00
4	00001	100	10 00

2. Binary

The binary encoding represents an integer by the binary numeral system where each digit's place value is a power of 2, starting from the rightmost digit. Each set of conformations for a given amino acid a_i is encoded by a substring of binaries, using n bits so that $2^n \leqslant c_i$ (see Table I). We cannot use the Hamming weight to check if a bitstring is in F. The total length of the bitstring is $\sum_i \lceil \log_2 c_i \rceil$ (see Table II).

3. BU-binary

Block-unary encodings can be considered as a balance between the unary and binary encodings, being more tunable to the limitations of the hardware. Block-unary encodings contain blocks of size g, each with a binary encoding (see Table I). Each block is an element in a unary string, where each block can encode for g variables, and the rest of the bits in the other blocks are set to zero. The all-zero state denotes when the block is inactive. This allows each block to encode $2^n - 1 = g$, where n represents the number of bits in a block. The total length of the bitstring is $\sum_i \lceil \frac{c_i}{g} \rceil \lceil \log_2(g+1) \rceil$ (see Table II).

B. Compiling to different quantum hardware

To implement a parametrized quantum operation related to the cost Hamiltonian in a quantum circuit, we represent it as $e^{i\gamma H_{\text{cost}}}$ with real parameter γ, mapping the quantum variables to logical qubits. This mapping relies on the specific operations supported by the target quantum processor, called the native gate set. The conversion requires breaking down complex operations in the cost Hamiltonian into elementary gates from the native gate set. Different quantum hardware architectures vary in both connectivity and the available number of qubits. Platforms may be able to operate with many-qubit interactions but offer a lower number of qubits, e.g., the ion-trapped architecture [6,61], compared to platforms offering a higher number of qubits that may be constrained to two-qubit interactions, e.g., the superconducting circuit architecture [5].

In the latter case, one must decompose the many-qubit interactions into less connected gates. Compiling the multi-qubit interactions of the quantum algorithms analyzed here

TABLE II. Comparison of resources for the unary, binary, and BU-binary encoding based on the cardinality vector **C** from Eq. (1). The functions are valid for the side-chain conformation-based and coordinate-based lattice models. Details about correction terms $\varepsilon_{\text{binary}}$ and $\varepsilon_{\text{BU-binary}}$ can be seen in Appendix A.

Resource	Unary	Binary	BU-binary
Qubits	$\sum_i c_i$	$\sum_i \lceil \log_2 c_i \rceil$	$\sum_i \lceil \frac{c_i}{g} \rceil \lceil \log_2(g+1) \rceil$
k locality	2	$\lceil \log_2[\max(c_i)] \rceil + \lceil \log_2[\max_2(c_i)] \rceil$	$2\lceil \log_2(g+1) \rceil$
Interactions	$\sum_i c_i + \binom{\sum_i c_i}{2}$	$\sum_i \lceil \log_2 c_i \rceil + \binom{\sum_i \lceil \log_2 c_i \rceil}{2} + \sum_{i<j} \sum_{m=3}^{k} \binom{\lceil \log_2 c_i \rceil + \lceil \log_2 c_j \rceil}{m} - \varepsilon_{\text{binary}}$	$\sum_i \lceil \frac{c_i}{g} \rceil \lceil \log_2(g+1) \rceil + \binom{\sum_i \lceil \frac{c_i}{g} \rceil \lceil \log_2(g+1) \rceil}{2} + \binom{\sum_i \lceil \frac{c_i}{g} \rceil}{2} \sum_{m=3}^{2\lceil \log_2(g+1) \rceil} \binom{2\lceil \log_2(g+1) \rceil}{m} - \varepsilon_{\text{BU-binary}}$

into controlled-NOT (CNOT) and single-qubit gates gives rise to a quadratic overhead; see Appendix B for a detailed description. Other native gate sets and the availability of higher-order gates such as three-qubit gates [62] may reduce the depth of the quantum algorithms and required resources. In Sec. IV B, we present how many k-qubit operations appear in the cost Hamiltonians of the protein folding models and how many one- and two-qubit operations would be needed to decompose the higher-order operations given a limited native set.

During the compilation process, optimization techniques can be applied to improve the overall efficiency and performance of the quantum circuit. These techniques include gate merging, gate cancellation, and gate reordering to minimize the circuit depth. We assume all-to-all connectivity and do not consider the distance between qubits, which could lead to decoherence. However, if the hardware lacks all-to-all connectivity, a naive qubit routing procedure might demand an additional circuit depth of $O(n^3)$. To tackle this problem, a SWAP network can be employed, offering a quadratic reduction in circuit depth compared to naive routing, resulting in a linear increase in operations [63–65].

IV. RESOURCE TRADEOFFS

We generate instances of folding and packing problems with amino acid chain lengths N ranging from 3 to 100 amino acids for five models, described in Sec. II, to compare the quantum resources required. For each problem instance, we have calculated the resources needed to form the variational state for the corresponding cost Hamiltonian, depending on the encoding used: unary, binary, or BU-binary.

Figures 2 and 3 present the resources for the coordinate-based HP model on the two- and three-dimensional lattice, respectively. The number of resources depends on the size of the lattice one chooses to fold the amino acid sequence on. We calculate resources for all rectangular and cubic grids containing a total number of lattice sites between N and $1.5N$. This range yields approximately ten problem instances for each sequence length N. Therefore, the presented resources represent averages and include one standard deviation. The resources for turn-based lattice models directly depend on the number of amino acids N. Figures 4 and 5 illustrate the number of resources needed for encoding the two- and three-dimensional cases, respectively. The side-chain conformation-based model's resources depend on the number of considered choices for each side chain. We generate 2000 instances to account for different proteins. For each

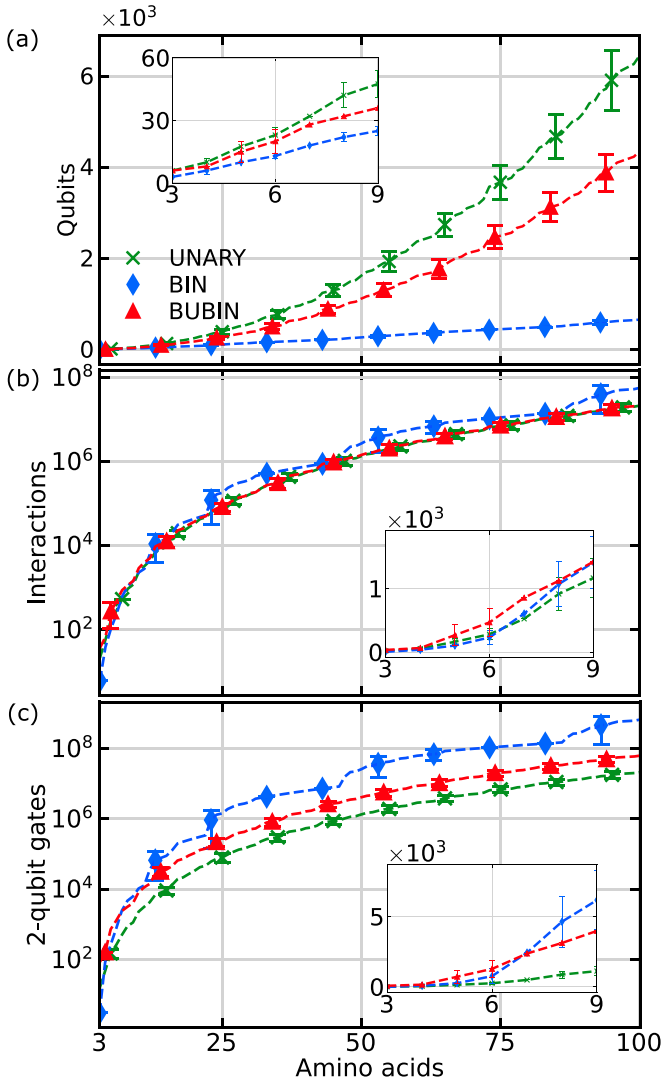

FIG. 2. Coordinate-based model on the square lattice. The number of (a) qubits, (b) interactions, and (c) two-qubit gates required to implement a parametrized quantum operation based on the cost Hamiltonian: $e^{-i\gamma H_{\text{cost}}}$, with H_{cost} given in Eq. (7), and a real parameter γ. We plot the required resources with unary (green cross), binary (blue rhombus), and BU-binary (red triangle) encodings as a function of the number of amino acids N. The problem instance size ranges from $N = 3, \ldots, 100$, and each figure inset zooms in on the results for fewer amino acids $N = 3, \ldots, 9$. For each N, we consider all rectangular grids with an area (number of sites) ranging from the lower bound N to the upper bound 50% larger than the lower bound, $1.5N$. The bars indicate one standard deviation.

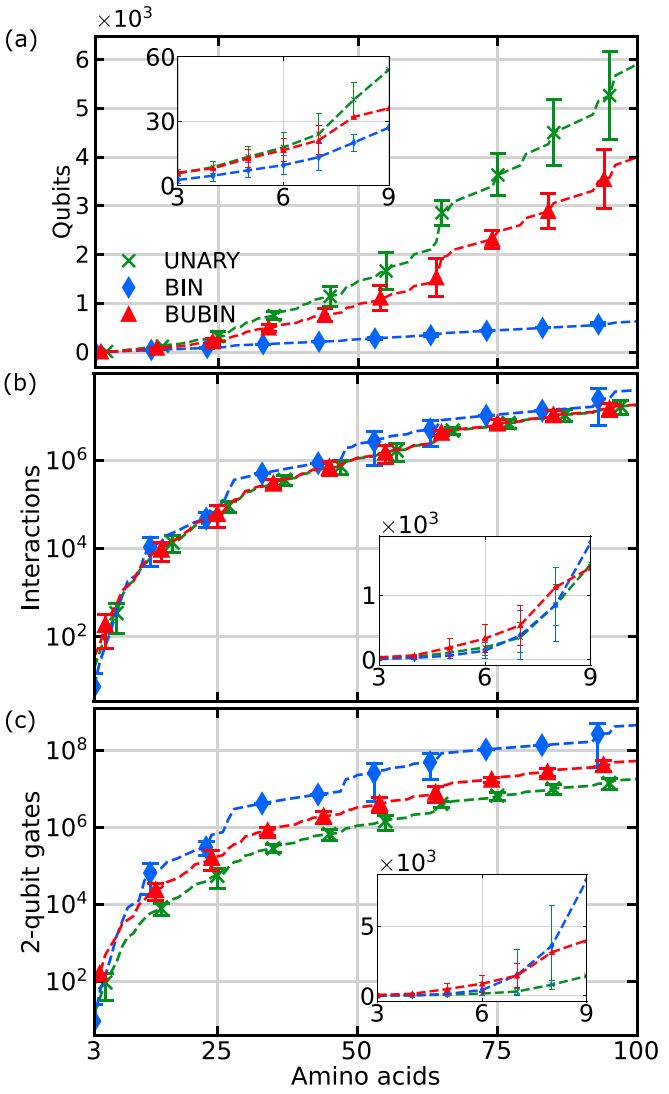

FIG. 3. Coordinate-based model on the cubic lattice. The number of (a) qubits, (b) interactions, and (c) two-qubit gates required to implement a parametrized quantum operation based on the cost Hamiltonian: $e^{-i\gamma H_{\text{cost}}}$, with H_{cost} given in Eq. (7), and a real parameter γ. We plot the required resources with unary (green cross), binary (blue rhombus), and BU-binary (red triangle) encodings as a function of the number of amino acids N. The problem instance size ranges from $N = 3, \ldots, 100$, and each figure inset zooms in on the results for fewer amino acids $N = 3, \ldots, 9$. For each N, we consider all cubic grids with a volume (number of sites) ranging from the lower bound N to the upper bound 50% larger than the lower bound, $1.5N$. The bars indicate one standard deviation.

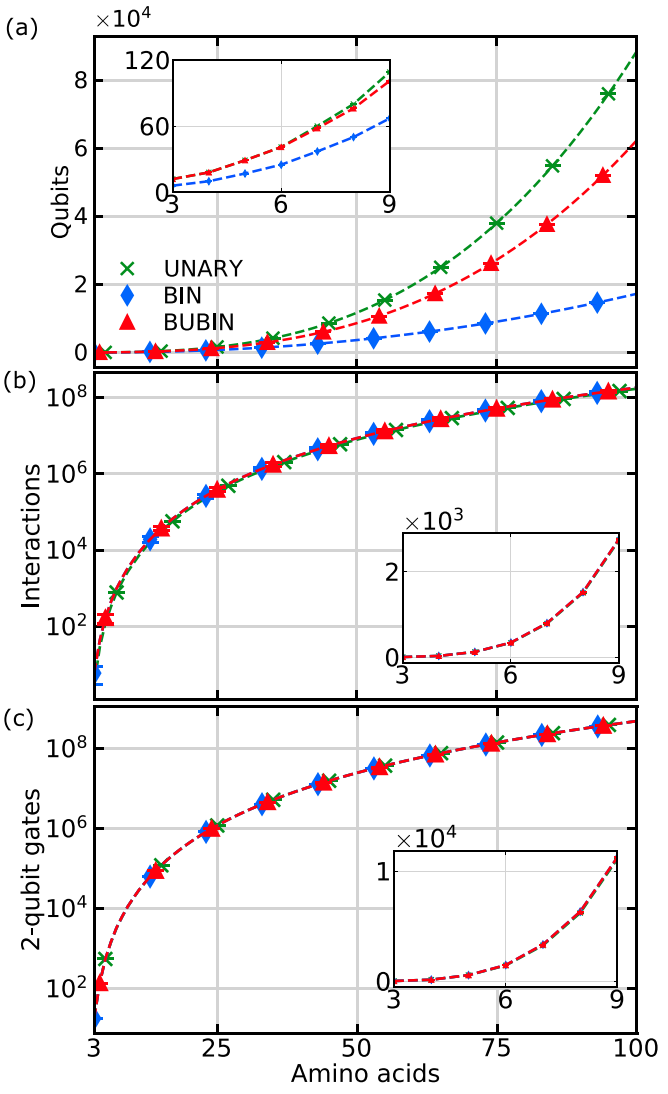

FIG. 4. Turn-based model on the square lattice. The number of (a) qubits, (b) interactions, and (c) two-qubit gates required to implement a parametrized quantum operation based on the cost Hamiltonian: $e^{-i\gamma H_{\text{cost}}}$, with H_{cost} given in Eq. (7), and a real parameter γ. We plot the required resources with unary (green cross), binary (blue rhombus), and BU-binary (red triangle) encodings as a function of the number of amino acids N. The problem instance size ranges from $N = 3, \ldots, 100$, and each figure inset zooms in on the results for fewer amino acids $N = 3, \ldots, 9$.

sequence length N, the number of conformations for each amino acid is drawn from a uniform distribution between 2 and 100. The average and standard deviation for the corresponding resources are shown in Fig. 6.

Furthermore, we compare our resource results to Ref. [13], where they consider a turn-based model on the tetrahedral lattice. The paper details the exact resource requirements of the number of qubits and interactions for up to 15 amino acids with fitted curves and presents the scaling of the number of terms in the Hamiltonian. Thus, we focus our comparison on the provided amino acid range of 15 and the scaling provided in the paper. Notably, we contrast the resource requirements of the tetrahedral lattice with the square lattice, due to their identical coordination numbers, despite their inherent differences.

In Sec. IV A, the analysis focuses on the number of qubits required for representing the protein structure as a bitstring for the different models given the three encodings. Section IV B studies the gates and many-body interactions needed, examining the computational complexity of implementing the various encoding schemes. Lastly, Sec. IV C investigates the size of the unfeasible solution set, examining the problem encoding efficiency and reduction of the search space, which may be related to the challenges of finding valid protein conformations within the given constraints.

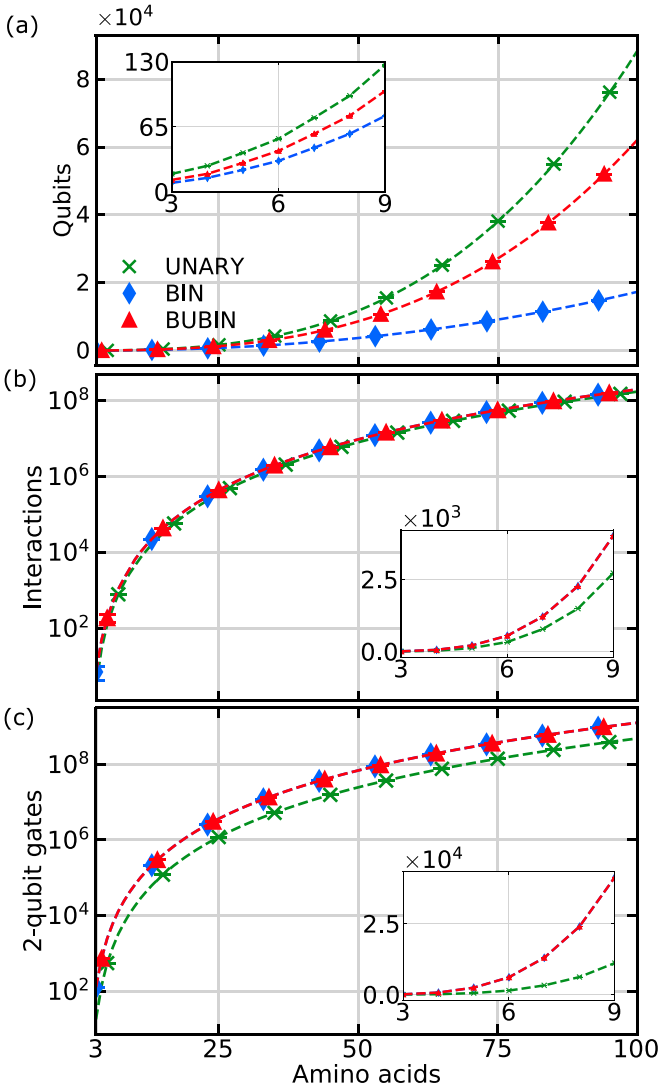

FIG. 5. Turn-based model on the cubic lattice. The number of (a) qubits, (b) interactions, and (c) two-qubit gates required to implement a parametrized quantum operation based on the cost Hamiltonian: $e^{-i\gamma H_{\text{cost}}}$, with H_{cost} given in Eq. (7), and a real parameter γ. We plot the required resources with unary (green cross), binary (blue rhombus), and BU-binary (red triangle) encodings as a function of the number of amino acids N. The problem instance size ranges from $N = 3, \ldots, 100$, and each figure inset zooms in on the results for fewer amino acids $N = 3, \ldots, 9$.

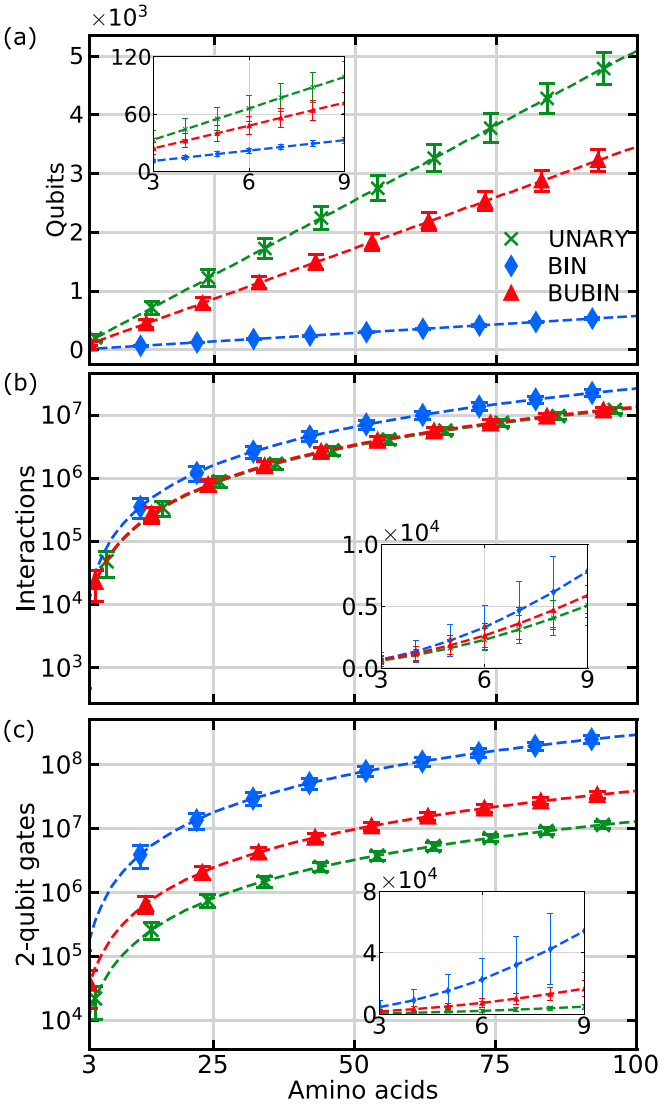

FIG. 6. Side-chain conformation-based model. The number of (a) qubits, (b) interactions, and (c) two-qubit gates required to implement a parametrized quantum operation based on the cost Hamiltonian: $e^{-i\gamma H_{\text{cost}}}$, with H_{cost} given in Eq. (7), and a real parameter γ. We plot the required resources with unary (green cross), binary (blue rhombus), and BU-binary (red triangle) encodings as a function of the number of amino acids N. The problem instance size ranges from $N = 3, \ldots, 100$, with the number of conformations for each amino acid uniformly distributed, $\mathcal{U}(2, 100)$. Each figure inset zooms in on the results for fewer amino acids $N = 3, \ldots, 9$ and a reduced number of conformations, $\mathcal{U}(2, 20)$. The bars indicate one standard deviation.

Here, we summarize our main findings based on the analysis. First, our calculations demonstrate that the minimum number of required qubits needed for encoding a protein with 100 amino acids in any of the considered models is around 600. This baseline is drawn by the binary encoding in the side-chain conformation-based and coordinate-based lattice model. Second, the unary encoding requires the highest number of qubits, but the binary encoding needs the most interactions and has a higher locality. The size of the unfeasible solution set—bitstrings corresponding to nonvalid solutions—is smaller for the binary encoding. The BU-binary encoding strikes a middle ground between the other two encodings. The block size variable g only shifts the BU-binary encoding closer to or further from the binary encoding with a larger or smaller g, respectively. For the sake of simplicity, we have used $g = 3$ throughout the paper. Third, the difference between folding on a two- or three-dimensional lattice is negligible compared to the total number of resources for sequence length N, both in the number of qubits and interactions. Finally, we note that the turn-based model on the tetrahedral lattice with the turn-based modeling of Ref. [13] is generally more resource efficient. Still, there are lattice sizes where the coordinate-based model with the binary encoding uses

the least amount of qubits and interactions to represent the same number of amino acids. Notably, the coordinate-based model has a higher k locality for the binary encoding than the turn-based model on the tetrahedral lattice. The opposite is true for the unary encoding, where the coordinate-based model has a lower k locality.

In the insets of the figures, we present the results for reduced chain lengths $N = 3, \ldots, 9$, and for the side-chain conformation-based model we reduce the maximum number of side-chain conformations to 20, averaging over 6000 instances for each sequence length N. Refer to Table II for an overview of the required resources calculated analytically based on the cardinality vector $\mathbf{C}$ from Eq. (1) for the coordinate-based model and side-chain conformation-based model. For a detailed breakdown of the required number of terms and auxiliary qubits as a function of sequence length N for the turn-based lattice model, please refer to Appendix C.

A. Number of qubits

The subfigures (a) of Figs. 2–6 show the number of qubits. The required qubits grow linearly with the number of amino acids for the side-chain conformation-based models, as seen in Fig. 6 and Table II. For the turn-based model, the number of required qubits is quadratic due to the required number of auxiliary qubits—needed to implement penalty terms mentioned in Sec. II A (see Fig. 4 and Appendix C). The coordinate-based lattice model may look linear with the number of amino acids as observed in Table II, but as seen in Figs. 2 and 3, the number of needed qubits is quadratic for the unary case and $ND \log N$ in the binary case. The nonlinear behavior comes from summing over the elements c_i of the cardinality vector $\mathbf{C}$ containing the lattice size, which, in turn, depends on the length of the amino acid sequence, as illustrated in Eq. (5). The sum equates to the number of amino acids times the number of qubits needed to encode each amino acid's position, N for the unary encoding and $D \log N$ for binary encoding.

The number of qubits for the unary encoding for the coordinate-based model and the conformation-based one is simply the sum of all elements in $\mathbf{C}$. The binary encoding reduces the number of qubits needed, compared to the unary encoding, to $\sum_i \lceil \log_2 c_i \rceil$. The advantage of the binary encoding becomes apparent in Figs. 2, 3, and 6. It requires nearly an order of magnitude fewer resources than the unary encoding. For the turn-based model, the improvement of using the binary encoding is not as significant (see Figs. 4 and 5). The BU-binary encoding lies in between the binary and unary encoding, and the total number of qubits needed is the sum of all the blocks used times the number of qubits per block. The number of blocks is inversely proportional to g, $\sum_i \lceil \frac{c_i}{g} \rceil$, and the number of qubits in a block is binary logarithmic in g, $\lceil \log_2(g+1) \rceil$. As the all-zero state in the blocks cannot be used to encode any information, each block encodes one less integer than the binary equivalent. As g approaches the upper limit equal to the largest element of $\mathbf{C}$, one obtains the binary encoding without possibly using the all-zero state. Using the lower limit $g = 1$ yields the unary encoding instead.

Comparing two- and three-dimensional models, the qubit difference is negligible relative to the total number of qubits required. For the coordinate-based model, the difference is only a few hundred because the minimum number of sites to allocate the amino acid chain is the same for two and three dimensions. The two-dimensional case has an average of more qubits as it is easier to pack the same number of sites tighter on a cubic lattice than on a square lattice. For the turn-based model, the number of needed qubits is similar because the number of possible directions in three dimensions causes two extra qubits per bead, which is insignificant compared to the number of auxiliary qubits needed.

For a 15-amino acid chain encoded with unary/binary on the tetrahedral lattice, 76/53 qubits are needed. Compared to the other two models with coordination number 4, the turn-based model requires 404/226 qubits, and the coordinate-based model needs an average of 135/52.5 qubits (SD = 15/7.5), with SD denoting standard deviation, for coordination number 4. The number of qubits required for the two models is similar, with the binary encoding making the coordinate-based model more resource efficient than the turn-based model on the tetrahedral lattice. If the smallest lattice sizes are used in the coordinate-based model, it outperforms the turn-based model on the tetrahedral lattice from 13 amino acids onwards. The scaling of the required number of qubits for the turn-based tetrahedral lattice [13] is comparable to the coordinate-based model with a qubit requirement scaling as N^2.

B. Gates and many-body interactions

The subfigures (b) of Figs. 2–6 indicate the number of terms in the cost Hamiltonian. The k locality of the Hamiltonian—specifying that the Hamiltonian terms act on at most k qubits—will depend crucially on the encoding. For specific hardware, the terms need to be decomposed into native gates. The higher the k locality of the terms, the more two-qubit gates are required. We have chosen native two-qubit CNOT gates as an example (see Appendix B), which is, e.g., relevant for the superconducting circuit platforms. The subfigures (c) show the number of two-qubit gates. Table II presents the analytical expressions for the number of gates and k locality for the coordinate-based lattice models and the side-chain conformation model. For details about the number of gates required for the turn-based lattice model, see Appendix C. The unary encoding in the coordinate-based and side-chain conformation-based models involves only one- and two-body interactions. Thereby, the curve for the unary encoding remains the same in subfigures (b) and (c) in Figs. 2, 3, and 6. The number of one-body interactions with the unary encoding is $\sum_i c_i$, with c_i the number of possible conformations for the ith amino acid. The number of two-body interactions is then given by $\binom{\sum_i c_i}{2}$. There are no higher-order interactions, that is, the Hamiltonian is two-local.

The binary encoding will have a higher k in the k locality than the unary encoding. For the coordinate-based and side-chain conformation-based models, the binary encoding k locality depends on the two largest elements c_i of the cardinality vector $\mathbf{C}$. Each amino acid a_i will need $\lceil \log_2 c_i \rceil$ qubits, and the two largest such values added together will yield the value of k. On average, the number of gates needed for the binary encoding is much larger than unary. To pairwise connect amino acids a_i and a_j, all the qubits associated with

the two amino acids, i.e., $\lceil \log_2 c_i \rceil + \lceil \log_2 c_j \rceil$, need to be connected, resulting in a m-body ($m \geqslant 3$) interaction requirement of $\binom{\lceil \log_2 c_i \rceil + \lceil \log_2 c_j \rceil}{m}$ for each pair. As explained in Sec. III, each classical binary variable x_i maps to $(1 - \sigma_i^z)/2$, and the multiplication of these binary variables leads to multiqubit terms of all orders. The total number of m-body interactions needed is then given by summing over all m and all pairs of amino acids. If $\lceil \log_2 c_i \rceil \geqslant 3$, we count higher-order interactions that are not unique to the pair and are present in every pair that includes a_i. These interactions could be grouped together and subtracted with a correction term $\varepsilon_{\text{binary}}$, as seen in Appendix A. The number of one-body interactions coincides with the number of qubits $\sum_i \lceil \log_2 c_i \rceil$. The number of two-body interactions is $\binom{\sum_i \lceil \log_2 c_i \rceil}{2}$, that is, all combinations of two qubits. The highest contribution to the number of gates for the binary encoding will be the double sum over all amino acids and all higher-order interactions $m = \{3, \ldots, k\}$.

In the BU-binary encoding, the k locality of the Hamiltonian is set with the integer variable g, which means that for a given hardware with k-body gates, it is possible to set g to match the hardware. Similar to the binary encoding, the number of one-body interactions in the BU-binary encoding is the same as the number of qubits and is given by $\sum_i \lceil \frac{c_i}{g} \rceil \lceil \log_2(g+1) \rceil$. The number of two-body interactions is equal to the number of pairwise combinations of all qubits $\binom{\sum_i \lceil \frac{c_i}{g} \rceil \lceil \log_2(g+1) \rceil}{2}$. Analogously to the binary encoding, the number of m-body interactions needed is calculated by considering pairwise interactions of blocks, as described in Sec. III A, and summing over all m-body terms that contribute to a pair of blocks, written as $\binom{\sum_i \lceil \frac{c_i}{g} \rceil}{2} \sum_{m=3}^{2\lceil \log_2(g+1) \rceil} \binom{2\lceil \log_2(g+1) \rceil}{m}$. Additionally, if $g \geqslant 4$, there will be interactions that appear multiple times and could be grouped, and again we subtract a small correction term $\varepsilon_{\text{BU-binary}}$, described in Appendix A.

For the turn-based model, the k locality of the interactions with the unary encoding is also lower than that of the binary encoding. However, in contrast to the other two models, this k locality of the unary encoding is now larger than 2. The number of gates needed for the unary is similar to, but still less than, the number of gates required for the binary and BU-binary encodings (see Figs. 4 and 5). Upon decomposition, the unary encoding demands even fewer gates than the binary encoding, especially evident in the three-dimensional case, as depicted in Fig. 5. Comparing Figs. 2 and 3 to Figs. 4 and 5, it is evident that the coordinate-based model requires fewer interactions than the turn-based model in both the two- and three-dimensional cases.

When examining the number of interactions in the turn-based model on both square and tetrahedral lattices, in agreement with prior research [13], it is evident that the latter model uses fewer interactions and has a lower locality. Our analysis shows that the coordinate-based model scales the same in the number of interactions as the turn-based model on the tetrahedral lattice. The coordinate-based model has a scaling of $O(N^4)$ (see Table II), and the turn-based model on the tetrahedral lattice has $O(N^4)$, as presented in Ref. [13]. Comparing the interactions needed for encoding a 15-amino acid chain in the coordinate-based model calls for a higher number of gates in general. For the unary encoding, the coordinate-based model has 9292.5 terms on average (SD = 2032.5), and the turn-based model on the tetrahedral lattice has 4997 terms. For the binary encoding, the coordinate-based model has 14 550 terms on average (SD = 9300), and the turn-based model on the tetrahedral lattice has 9994 terms. Again, there are smaller lattices where the coordinate-based model using the binary encoding calls for fewer resources.

TABLE III. Comparison of the relative size of the feasible solution set, $\frac{|F|}{|S|}$, for the unary, binary, and BU-binary encoding. These results are solely based on choosing one conformation per amino acid and thereby exact for the side-chain conformation-based model and approximate for the two lattice models.

Encoding	Relative size of feasible set
Unary	$\prod_i c_i / 2^{c_i}$
Binary	$\prod_i c_i / 2^{\lceil \log_2(c_i) \rceil}$
BU-binary	$\prod_i c_i / 2^{2\lceil \frac{c_i}{g} \rceil}$

C. Size of the unfeasible solution set

The feasible solution set F encompasses all bitstrings encoding valid physical solutions, where precisely one conformation is selected for each amino acid, and no constraints are violated. The choice of the encoding for the problem will affect the size of this feasible solution set compared to all possible solution bitstrings. Here, we discuss the ratio between the sizes of the feasible solution set F and the total solution set S, which contains the feasible and unfeasible solution sets, as an additional measure of the problem encoding efficiency. We aim to minimize the unfeasible solution set to reduce the search space. However, a smaller unfeasible solution set does not ensure ease in finding the optimal state, as the energy landscape can still be intricate and challenging to navigate. This landscape may exhibit barren plateaus or steep mountains, potentially leading to entrapment in local minima.

The bitstrings constituting the feasible solutions for the coordinate-based lattice model are determined by three penalty terms presented in Sec. II A. In Fig. 7, we show an approximate upper bound of the relative size of the feasible solution set for the coordinate-based lattice model. We have only considered two model constraints: one amino acid per site and one site per amino acid. The approximate feasible solution set F_* contains solutions that may have a broken chain. Similarly, we present an approximate upper bound of $|F_*|/|S|$ for the turn-based model in Fig. 8, as the calculations are only based on the cardinality vector in Eq. (5) and do not contain the overlap constraint. The auxiliary qubits of the turn-based lattice model are not accounted for in this section, as they do not affect the feasibility of the bitstrings. For the side-chain conformation-based model, we present the exact results in Fig. 9. Table III includes the analytical expression of the relative sizes of the feasible solution sets. The calculations are based on the bitstrings generated by choosing just one conformation per amino acid for the three encodings.

The relative size of the feasible solution set decreases rapidly with the increasing number of amino acids for all

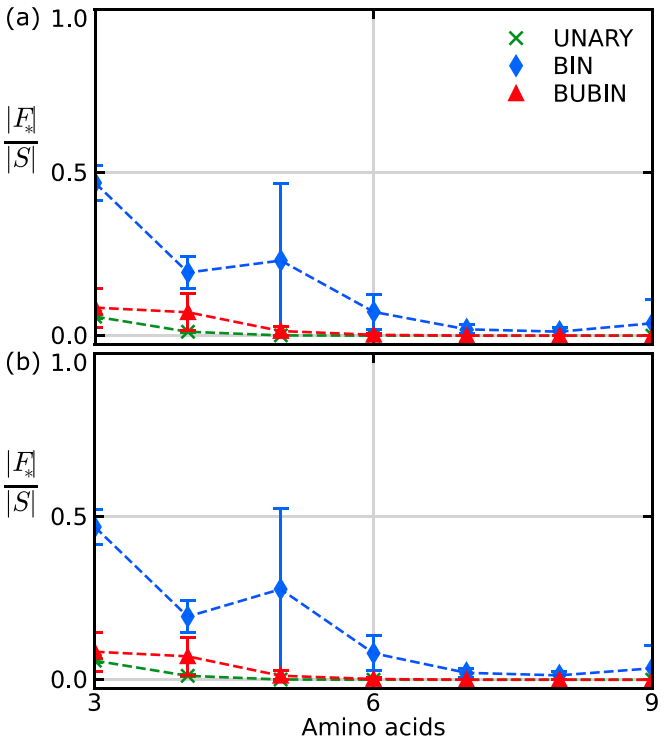

FIG. 7. Coordinate-based model on square (a) and cubic (b) lattices. Ratio between the size of the approximate feasible solution set $|F_*|$ and the number of possible bitstrings $|S|$ as a function of the number of amino acids N for the unary (green cross), binary (blue rhombus), and BU-binary (red triangle) encodings. For each N, we consider all square/cubic grids with an area/volume (number of sites) ranging from the lower bound N to the upper bound 50% larger than the lower bound, $1.5N$. The bars indicate one standard deviation.

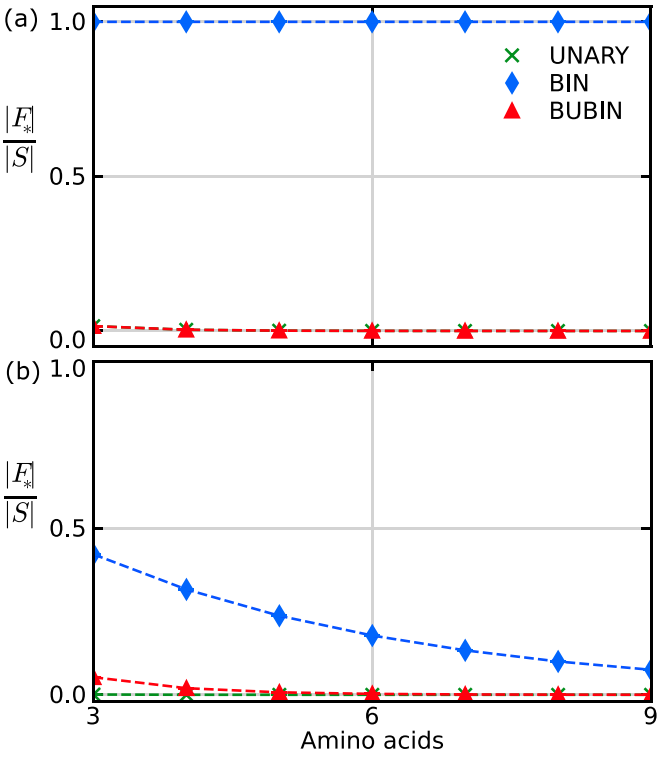

FIG. 8. Turn-based model on square (a) and cubic (b) lattices. Ratio between the size of the approximate feasible solution set $|F_*|$ and the number of possible bitstrings $|S|$ as a function of the number of amino acids for the unary (green cross), binary (blue rhombus), and BU-binary (red triangle) encodings.

encodings. The turn-based model on the two-dimensional lattice stands out as the only exception, with a relative size exceeding 25% from nine amino acids onwards, as shown in Fig. 8. The unary encoding will have a high number of unfeasible solutions, as we need to keep Hamming weight 1 for each block of c_i qubits, with c_i the elements of the vector in Eq. (1). Thereby, for the unary encoding, the ratio $|F|/|S|$ tends to zero as the number of conformations per amino acid increases. It is upper bounded by 2^{-N}, where each amino acid only has two conformations.

In contrast, the binary encoding will generally have a smaller unfeasible solution set than the unary encoding, even if its size depends highly on the structure of the problem instance. In the best-case scenario for the binary encoding, the unfeasible solution set is empty when each amino acid has $2^l, l \in \mathbb{N}$ conformations. The turn-based model in two dimensions is an example of a best-case scenario of the binary encoding, with each amino acid having $2^2 = 4$ conformations (see Fig. 8). In the worst-case scenario for the binary encoding, when each amino acid has $2^l + 1$ conformations, the unfeasible solution set tends to zero as with the unary representation. Both the worst case and the general case of the binary encoding are better than the unary encoding, though, as seen in Fig. 9. While the restriction of only using two-level systems applies to specialized hardware such as quantum annealers, a gate-based quantum computer theoretically permits working with qudits of any dimension d, leading to a best-case scenario of d^l conformations. Again, the BU-binary encoding falls between the unary and binary encodings, but

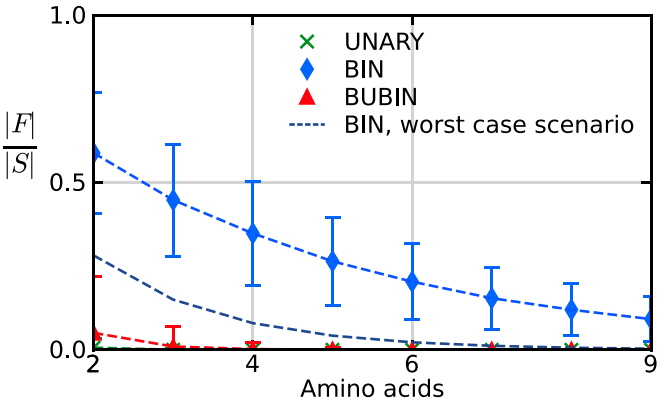

FIG. 9. Side-chain conformation-based models. Ratio between the size of the feasible solution set $|F|$ and the number of possible bitstrings $|S|$ as a function of the number of amino acids N for the unary (green cross), binary (blue rhombus), and BU-binary (red triangle) encodings. The problem instance size ranges from $N = 3, \ldots, 9$, with the number of conformations for each amino acid uniformly distributed, $\mathcal{U}(2, 20)$. The bars indicate one standard deviation. The dashed blue line without a marker is the worst-case scenario for the binary encoding.

the relative size of the feasible set rapidly approaches zero, as seen in Figs. 7–9, since multiple blocks are required to encode conformations for each amino acid.

V. CONCLUSION AND OUTLOOK

In summary, we have analyzed the resources for five coarse-grained models: HP models on the lattice, with a turn-based and coordinate-based representation of the amino acids locations, and the off-lattice side-chain packing model. In particular, we have computed the required qubits, interactions, and two-qubit gates for problem formulations associated with three encodings: unary, binary, and BU-binary.

We conclude that current NISQ devices are unsuitable for simulating protein folding instances larger than a proof of concept due to the significant gate requirement. The number of gates needed to address average-sized human proteins reaches at least 10^7, which is significantly greater than what is now possible with current gate fidelities of 99.9% [66,67]. However, the necessary number of qubits is attainable within the NISQ era [68], with the side-chain packing problem and protein folding using coordinate-based lattice models being the most feasible applications. For instance, when employing the binary encoding, a chain of 100 amino acids can be represented with fewer than 1000 qubits on average.

When comparing the resource requirements of the three encodings, we observe that using the binary representation calls for the least number of qubits and yields a smaller unfeasible solution set, but translates to the highest-order interactions in the Hamiltonian. The binary encoding is thereby more suitable for quantum hardware with access to multiqubit gates, like ion-trap computers. Conversely, hardware limitations for the k locality of the quantum gates, such as with superconducting qubits, favor using the unary encoding, even if it requires more qubits to represent the same problem instance. The BU-binary encoding, with the flexibility of choosing the block size, can strike a balance between the unary and binary encodings to accommodate specific device limitations. Our conclusions echo earlier results analyzing encodings for quantum simulations of quantum models [47].

Counterintuitively, we find that the coordinate-based model is more resource efficient than the turn-based model for the square and cubic lattice, both in time and space complexity. Even if the turn-based model requires fewer qubits for encoding protein conformations, we need additional auxiliary qubits to prevent overlapping conformations and encode pairwise interactions. These additional qubits significantly increase the qubit requirement by one order of magnitude compared to the coordinate-based model. Further, the improved turn-based model on the tetrahedral lattice [13] scales as the coordinate-based model on the square lattices in the number of qubits and gates needed. For shorter chains with a maximum of 15 amino acids, the turn-based model on the tetrahedral lattice generally uses fewer resources. Still, the coordinate-based model with the unary encoding has a lower k locality than the tetrahedral model, and with the binary encoding it requires fewer qubits and gates for small lattices.

Even with a resource-efficient model, using a hybrid quantum algorithm does not guarantee finding the native structure of the protein. As presented by previous work [12,34,39], the more challenging part of achieving a quantum advantage in protein folding simulations may be the classical optimization of the parameters in the quantum circuit. These obstacles generate skepticism about QAOA's ability to address the protein folding problem in the near future [39]. Research in optimizing the quantum circuit goes hand in hand with the search for resource-efficient model formulations. A potential approach to address the requirement for auxiliary qubits could involve applying unequal penalization to the inequality constraints [69]. We hope our paper will spur exploration for more resource-efficient models for protein folding and stimulate further research into qualitatively better quantum computing systems.

The code used in this paper can be found in Ref. [70]. All calculations are performed in PYTHON with NUMPY [71] and SCIPY [72], and all the plots are generated with MATPLOTLIB [73].

ACKNOWLEDGMENTS

We thank Anders Irbäck, Lucas Knuthson, Sandipan Mohanty, and Carsten Peterson for the essential discussions that made this paper what it is. We thank Ivano Tavernelli for helpful discussions and for sharing details of his paper's results. We acknowledge support from the Knut and Alice Wallenberg Foundation through the Wallenberg Center for Quantum Technology. L.G.-Á. further acknowledges funding under Horizon Europe programme HORIZON-CL4-2022-QUANTUM-01-SGA via the project 101113946 OpenSuperQPlus100.

APPENDIX A: CORRECTION TERM FOR BINARY AND BU-BINARY ENCODING

In Sec. IV B of the main text, we calculate the number of interactions in the cost Hamiltonian of the side-chain conformation-based and coordinate-based lattice models. To compute the number of higher-order Hamiltonian terms for the binary encoding, we consider all pairwise amino acid interactions and all combinations of multiqubit terms ranging from order 3 to the total number of qubits associated with each amino acid pair. We need all the terms because, as explained in Sec. III, each classical binary variable x_i maps to $(1-\sigma_i^z)/2$, and the multiplication of these binary variables leads to multiqubit terms of all orders. An amino acid a_i has c_i possible conformations, as shown in Eq. (1). If the amino acid a_i is encoded by at least three qubits, a binomial term accounting for all possible m-element qubit combinations (with $m \geqslant 3$) will include some terms that only involve qubits encoding the amino acid a_i. However, we only want to count these m-element terms once, and therefore, we need to subtract them for every pairwise interaction in which the amino acid a_i is included except one.

We can group these extra multiqubit interactions in correction terms and subtract them from the total number. The correction terms are given by

$$\varepsilon_{\text{binary}} = \sum_{i}^{N} \sum_{m=3}^{\lceil \log_2(c_i) \rceil} \binom{\lceil \log_2(c_i) \rceil}{m} (N-2), \tag{A1}$$

for the binary encoding, where we assume that $N > 2$.

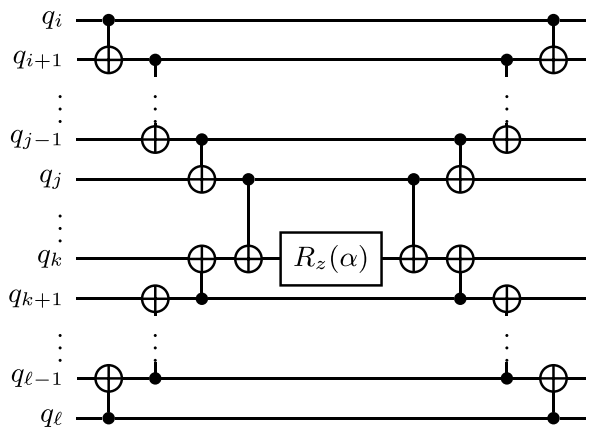

FIG. 10. A circuit based on CNOT ladders implementing the Pauli tensor evolution in Eq. (B1) in terms of two-qubit interactions.

For the BU-binary encoding, instead of amino acids, we consider all pairwise block interactions. By an analogous argument, the correction term is given by

$$\varepsilon_{\text{BU-binary}} = \left(-2 + \sum_i^N \left\lceil \frac{c_i}{g} \right\rceil\right) \sum_{m=3}^{\lceil \log_2(g+1)\rceil} \binom{\lceil \log_2(g+1)\rceil}{m}. \tag{A2}$$

Note that for the number of qubits that encode one block to be at least 3, we need $\lceil \log_2(g+1)\rceil \geqslant 3$, or $g \geqslant 4$. In this paper we used $g = 3$, so $\varepsilon_{\text{BU-binary}} = 0$. In general, both correction terms are small compared to the number of interactions.

APPENDIX B: COMPILATION OF MULTIQUBIT GATES INTO UNIVERSAL GATE SETS

In Sec. IV of the main text, we calculate the number of two-qubit interactions in each of the five models, together with the encodings. To transition from k-body interaction to two-body interactions, we decompose the higher-order gates as follows. The phase gadgets are a class of unitaries related to Pauli tensor evolution operators, essential blocks in variational quantum algorithms. The multiqubit interactions relevant to the quantum algorithms analyzed in this paper can be implemented in a superconducting circuit processor using CNOT-staircase constructions. In particular, we consider the general operation

$$U_Z^{ijk\ell}(\alpha) = \exp\left[-i\frac{\alpha}{2}\sigma_z^i \cdots \sigma_z^j \sigma_z^k \cdots \sigma_z^\ell\right)\Big], \tag{B1}$$

which can be implemented with CNOT ladders, as shown in Fig. 10. With such construction, one needs $2(m-1)$ two-qubit gates for an m-body Pauli tensor, i.e., an interaction term acting upon m qubits. Here, we do not consider reductions due to successive applications of these blocks or overheads for two-qubit gates between distant qubits.

This well-known CNOT-staircase construction is particularly suitable for quantum hardware with a gate set comprising single- and two-qubit operations, such as superconducting circuits. The final gate count depends on the particular available gate set and compilation. For instance, access to multiqubit gates [74,75] and controlled arbitrary-phase gates CZ_ϕ reduces the algorithm depth [76]. Moreover, the particular qubit connectivity of the device introduces a gate overhead due to routing steps comprising additional SWAP gates that allow distant qubits to interact.

APPENDIX C: LOCALITY AND INTERACTION IN THE TURN-BASED LATTICE MODEL

In Sec. IV of the main text, we calculate the number of qubits and interactions of the turn-based model on the square and cubic lattice. Here, we review previous work on the turn-based lattice model. We pay attention to the locality of terms and count the interactions associated with each equation. In Ref. [35], the authors construct a cost Hamiltonian which encodes a two-dimensional ($D = 2$) HP-lattice model with the binary encoding. In Ref. [36], the authors expand the earlier model to three dimensions ($D = 3$) using the MJ energies, also with the binary encoding.

1. Cost Hamiltonian

The cost Hamiltonian has a general form consisting of three terms: one term that penalizes the choice of two consecutive turns folding back onto itself H_{back}, one term that penalizes overlapping later in the amino acid chain H_{overlap}, and H_{pair} which is is the two-body interaction term (HP or MJ potential).

2. Taking a turn

To begin, some equations are designed to represent turns in specific spatial directions. These equations evaluate as TRUE if the jth turn occurs in the specified spatial direction, such as the x direction:

$$d_{+x}^j = (1 - q_{3j-4})\, q_{3j-5} q_{3j-3}, \tag{C1}$$

$$d_{-x}^j = (1 - q_{3j-3})\, q_{3j-5} q_{3j-4}. \tag{C2}$$

These equations are then used to calculate how many times each turn has occurred in the previous chain, thereby getting the coordinate of the amino acid of interest. In the unary encoding, the turn equation for each direction only includes one qubit, as noted earlier [12]. Each equation has a locality of the lattice dimension D in the binary and BU-binary case and one in the unary case.

3. Not turning back

The term H_{back} uses AND functions ($\wedge$) that take two turns as input and return true if the second goes backward. For example, to penalize turning right, $+x$, at turn j AND then left, $-x$, in turn $j+1$, we have the circuit

$$\begin{aligned} d_{+x}^j \wedge d_{-x}^{j+1} &= [(1 - q_{3j-4}) q_{3j-5} q_{3j-3}] \\ &\quad \times [(1 - q_{3j}) q_{3j-2} q_{3j-1}] \\ &= (1 - q_{3j-4} - q_{3j} + q_{3j} q_{3j-4}) \\ &\quad \times q_{3j-5} q_{3j-2} q_{3j-3} q_{3j-1}. \end{aligned} \tag{C3}$$

To penalize turning right then left OR ($\vee$) left then right, we have

$$\begin{aligned}&\left(d^j_{+x}\wedge d^{j+1}_{-x}\right)\vee\left(d^j_{-x}\wedge d^{j+1}_{+x}\right)\\&\quad=\left(d^j_{+x}\wedge d^{j+1}_{-x}\right)+\left(d^j_{-x}\wedge d^{j+1}_{+x}\right).\end{aligned}\tag{C4}$$

By combining all versions of right then left, left then right, and so on,

$$\left(d^j_{+x}\wedge d^{j+1}_{-x}\right)\vee\left(d^j_{-x}\wedge d^{j+1}_{+x}\right)\vee\left(d^j_{+y}\wedge d^{j+1}_{-y}\right)\vee\left(d^j_{-y}\wedge d^{j+1}_{+y}\right),\tag{C5}$$

it is possible to penalize all backturns. The H_{back} results in at most two-dimensional-local terms in the binary and BU-binary encodings, and two-local terms in the unary encoding. The final H_{back} is then given by

$$\begin{aligned}H_{\text{back}}=\lambda_{\text{back}}\Bigg\{&\left(q_0\vee d^2_{-x}\right)+\left[(1-q_0)\vee d^2_{-y}\right]\\&+\sum_{j=2}^{N-3}\left[\left(d^j_{+x}\wedge d^{j+1}_{-x}\right)+\left(d^j_{-x}\wedge d^{j+1}_{+x}\right)\right.\\&+\left(d^j_{+y}\wedge d^{j+1}_{-y}\right)+\left(d^j_{-y}\wedge d^{j+1}_{+y}\right)\\&\left.+\left(d^j_{+z}\wedge d^{j+1}_{-z}\right)+\left(d^j_{-z}\wedge d^{j+1}_{+z}\right)\right]\Bigg\}.\end{aligned}\tag{C6}$$

The number of interactions on the three-dimensional lattice is $(N-5)$, from the summation, times the terms in the AND expressions, which is 4×2 for the x axis and z axis and 12×2 for the y axis and sums to 20 terms. The number of interactions on the two-dimensional lattice is $2N-10$, from the summation, times 11.

4. Position of the amino acids

Further, we need equations to describe the position of amino acid i with the help of the boolean turn equations above, e.g., in the x direction, we have

$$x_i=\begin{cases}0, & \text{if } i=0\\ & (a_0 \text{ occupy the origin}),\\ 1+q_0+ & \\ \sum_{j=2}^{i-1}\left(d^j_{+x}-d^j_{-x}\right), & \text{otherwise},\end{cases}\tag{C7}$$

which summarizes the left and right turns of previous amino acids to determine which x coordinate the amino acid has. The locality of the position equations is lower than for the turn equations; the minus sign inside the summation in x_i cancels out the highest-local term, resulting in, at most, next-to-highest-local terms in the 3D case, and we get

$$\begin{aligned}d^j_{+x}-d^j_{-x}&=(1-q_{3j-4})q_{3j-5}q_{3j-3}\\&\quad-(1-q_{3j-3})q_{3j-5}q_{3j-4}\\&=q_{3j-3}q_{3j-5}-q_{3j-3}q_{3j-4}q_{3j-5}\\&\quad-q_{3j-4}q_{3j-5}+q_{3j-3}q_{3j-4}q_{3j-5}\\&=q_{3j-3}q_{3j-5}-q_{3j-4}q_{3j-5},\end{aligned}\tag{C8}$$

which also holds for y_i and z_i. This is repeated in the two-dimensional case where two-local equations for the position are reduced to one-local. The position equations have the locality of $D-1$ in binary and BU-binary encodings and a locality of 1 in the unary encoding.

TABLE IV. Qubits needed to encode the information on the distance between two amino acids for each encoding, that is, the variable u in Eq. (C13).

Encoding	Qubits (u)
Unary	$(j-i)$
Binary	$\log_2(j-i)$
BU-binary	$\lceil(j-i)/g\rceil\lceil\log_2(g+1)\rceil$

5. Distance between amino acids

Moreover, we need equations that give the distance between two amino acids j and k:

$$D_{ij}=(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2.\tag{C9}$$

All terms in the distance equation are $2(D-1)$-local for the binary and BU-binary encodings, and two-local for the unary encoding, as all other ordered terms cancel out. The number of interactions in the distance equation will be the sum of all possible pairs of the terms in the position equations. The terms for the maximal distance N will contain the other distance terms, and the maximal number of terms is thus $\binom{N}{2}$.

6. Avoiding overlap

Auxiliary qubits are introduced to enforce the bounds on the distance function

$$0\leqslant D_{ij}\leqslant(i-j)^2,\tag{C10}$$

and to ensure no amino acids overlap,

$$D_{ij}\neq 0,\ \text{if } i>j+3,\tag{C11}$$

thereby enforcing the inequality constraints

$$D_{ij}\geqslant 1.\tag{C12}$$

The total number of auxiliary qubits needed to encode the information of the distance between amino acids for all amino acid pairs in the binary encoding is

$$M^{\text{dist}}_{\text{aux}}=\sum_{i=0}^{N-5}\sum_{j=i+4}^{N-1}\lceil 2u(|1+i-j|\bmod 2)\rceil,\tag{C13}$$

where the number of qubits u to encode the information will differ between the encodings according to Table IV.

We introduce the slack variables α_{ij} from the qubit index $DN-7$ onward. Each slack variable is encoded on μ_{ij} qubits, with the qubit pointer given by

$$p_{ij}=(DN-8)+\sum_{u=0}^{i}\sum_{n=u+4}^{N-1}\mu_{un}-\sum_{m=j}^{N-1}\mu_{im}.\tag{C14}$$

We can thus write the slack variables as

$$\alpha_{ij}=\sum_{j=0}^{\mu_{ij}-1}q_{p_{ij}+j}2^{\mu_{ij}-1-j}.\tag{C15}$$

TABLE V. The number of interactions and type of interaction for each operator in the double sum in Eq. (C17). Here, we denote the dimensionality of the lattice D, the number of amino acids N, the number of auxiliary qubits for encoding the distance $M_{\text{aux}}^{\text{dist}}$, the equation for distance between two amino acids D_{ij} from Eq. (C9), the slack variables' value α_{ij} from Eq. (C15), and the number of qubits needed to store the slack variables μ_{ij}.

Operator	Interactions	Interacting qubits	k-locality binary and BU-binary	k-locality unary
$2^{\mu_{ij}2} - 2^{\mu_{ij}+1}$	0			
D_{ij}^2	$\binom{\binom{N}{2}}{2}$	$q_{0,\ldots,(DN-8)}$	$4(D-1)$	4
α_{ij}^2	$\binom{M_{\text{aux}}^{\text{dist}}}{2}$	$q_{(DN-7),\ldots,M_{\text{aux}}^{\text{dist}}}$	2	2
$2D_{ij}\alpha_{ij}$	$\binom{N}{2}M_{\text{aux}}^{\text{dist}}$	$q_{0,\ldots,M_{\text{aux}}^{\text{dist}}}$	$2(D-1)+1$	3
$-2^{\mu_{ij}+1}\alpha_{ij}$	$M_{\text{aux}}^{\text{dist}}$	$q_{(DN-7),\ldots,M_{\text{aux}}^{\text{dist}}}$	1	1

The final Hamiltonian to avoid the overlap

$$H_{\text{overlap}} = \lambda_{\text{overlap}} \sum_{i=0}^{N-5} \sum_{j=i+4}^{N-1} (|1+j-i| \bmod 2) \times (2^{\mu_{ij}} - D_{ij} - \alpha_{ij})^2, \tag{C16}$$

where μ_{ij} is an integer. Hence, we calculate the needed qubits by counting the terms linked to the following operators except the first one:

$$(2^{\mu_{ij}} - D_{ij} - \alpha_{ij})^2 = 2^{\mu_{ij}2} - 2^{\mu_{ij}+1} + D_{ij}^2 + \alpha_{ij}^2 + 2D_{ij}\alpha_{ij} - 2^{\mu_{ij}+1}\alpha_{ij}. \tag{C17}$$

See Table V for an overview of the k locality and interacting qubits associated with the previous operators. The number of interactions in the term D_{ij}^2 is all combinations of connecting two amino acids, and then all combinations of connecting these combinations 2 and 2:

$$\binom{\binom{N}{2}}{2} = \frac{1}{8}(N^4 - 2N^3 - N^2 + 2N) \propto N^4, \tag{C18}$$

and the locality is $4(D-1)$ for the binary and BU-binary encodings, and 4 for the unary encoding. The number of interactions in the α_{ij}^2 is all combinations of connecting the auxiliary qubits $M_{\text{aux}}^{\text{dist}}$ pairwise. The number of interactions in the term $2D_{ij}\alpha_{ij}$ is the number of interactions in D_{ij} times the number of terms in α_{ij}, which is equal to $M_{\text{aux}}^{\text{dist}}$. The last term $2^{\mu_{ij}+1}\alpha_{ij}$ has locality 1.

7. Pairwise amino acid interaction

The complexity of the Hamiltonian H_{pair} in the turn-based model arises from the need for encoding distance information between amino acids to calculate their lattice interactions. This requires circuit-based calculations, demanding additional qubits to track interactions between specific amino acids (j and k) on the lattice:

$$\omega_{jk} = \begin{cases} 1, & \text{if } D_{jk} = 1 \\ 0, & \text{otherwise,} \end{cases} \tag{C19}$$

and the number of auxiliary qubits needed is

$$M_{\text{aux}}^{\text{pair}} = \sum_{j=0}^{N-4} \sum_{k=j+3}^{N-1} [(|j-k|) \bmod 2]. \tag{C20}$$

The auxiliary qubits can then be used with an energy matrix (either MJ or HP potential) P_{jk}, and the expression on the pairwise Hamiltonian is formulated as

$$\begin{aligned} H_{\text{pair}} &= \sum_{j=0}^{N-4} \sum_{k=j+3}^{N-1} (|j-k| \bmod 2)\omega_{jk}P_{jk}(2 - D_{jk}) \\ &= \sum_{j=0}^{N-4} \sum_{k=j+3}^{N-1} (|j-k| \bmod 2)P_{jk}(2\omega_{jk} - D_{jk}\omega_{jk}). \end{aligned} \tag{C21}$$

The number of interactions to build the pair Hamiltonian is thereby given by the number of terms in D_{ij} times the number of auxiliary qubits:

$$K^{\text{pair}} = \binom{N}{2} M_{\text{aux}}^{\text{pair}}. \tag{C22}$$

8. Total number of interactions

We get the total interaction for the turn-based lattice model:

$$\begin{aligned} K^{\text{total}} = {} & (DN-8)c_1 \\ & + \binom{\binom{N}{2}}{2} + \binom{M_{\text{aux}}^{\text{dist}}}{2} + \binom{N}{2} M_{\text{aux}}^{\text{dist}} \\ & + \binom{N}{2} M_{\text{aux}}^{\text{pair}}, \end{aligned} \tag{C23}$$

where the constant c_1 is dependent on the lattice dimension and is small in comparison to the other terms.

9. Total number of qubits

The total number of qubits is given by

$$M_{\text{total}} = M_{\text{conf}} + M_{\text{aux}}^{\text{dist}} + M_{\text{aux}}^{\text{pair}}. \tag{C24}$$

For the binary encoding this is

$$\begin{aligned} M_{\text{total}}^{\text{binary}} = {} & (DN - c_1) \\ & + \sum_{i=0}^{N-5} \sum_{j=i+4}^{N-1} \lceil 2\log_2(j-i) \rceil (|1+j-i| \bmod 2) \\ & + \sum_{j=0}^{N-4} \sum_{k=j+3}^{N-1} (|j-k| \bmod 2), \end{aligned} \tag{C25}$$

but here, we changed the order of i and j in the logarithm, as j is always larger than i, and we cannot take the logarithm of a negative value. Asymptotically, the number of qubits grows as

$$M_{\text{total}}^{\text{binary}} = \frac{1}{2}N^2 \log_2 N - \left(\frac{1}{4} - \frac{3}{4\ln 2}\right)N^2 + O(N). \quad \text{(C26)}$$

Next, the total number of qubits needed for the unary encoding is then

$$M_{\text{total}}^{\text{unary}} = 2DN + \sum_{i=0}^{N-5}\sum_{j=i+4}^{N-1} \lceil 2(j-i)\rceil (|1+j-i| \bmod 2) + \sum_{j=0}^{N-4}\sum_{k=j+3}^{N-1} (|j-k| \bmod 2). \quad \text{(C27)}$$

In this case, the number of qubits grows asymptotically as

$$M_{\text{total}}^{\text{unary}} = \tfrac{1}{6}N^3 + \tfrac{1}{4}N^2 + O(N). \quad \text{(C28)}$$

Lastly, the total number of qubits needed for the BU-binary encoding is

$$M_{\text{total}}^{\text{BU-binary}} = \lceil \log_2 (g+1)\rceil \lceil ND/g\rceil + \sum_{i=0}^{N-5}\sum_{j=i+4}^{N-1} 2\lceil (j-i)/g\rceil \lceil \log_2 (g+1)\rceil (|1+j-i| \bmod 2) + \sum_{j=0}^{N-4}\sum_{k=j+3}^{N-1} (|j-k| \bmod 2). \quad \text{(C29)}$$

This number scales asymptotically as

$$M_{\text{total}}^{\text{BU-binary}} = \frac{\log_2(1+g)}{6g}N^3 + \frac{\log_2(1+g)}{4g}N^2 + O(N). \quad \text{(C30)}$$

[1] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. **26**, 1484 (1997).

[2] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96 (Association for Computing Machinery, New York, 1996), pp. 212–219.

[3] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, Layered architecture for quantum computing, Phys. Rev. X **2**, 031007 (2012).

[4] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[5] M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, and W. D. Oliver, Superconducting qubits: current state of play, Annu. Rev. Condens. Matter Phys. **11**, 369 (2020).

[6] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, Trapped-ion quantum computing: Progress and challenges, Appl. Phys. Rev. **6**, 021314 (2019).

[7] H.-S. Zhong *et al.*, Quantum computational advantage using photons, Science **370**, 1460 (2020).

[8] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, Nat. Rev. Phys. **3**, 625 (2021).

[9] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv:1411.4028.

[10] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature (London) **549**, 242 (2017).

[11] A. Caunhye, X. Nie, and S. Pokharel, Optimization models in emergency logistics: A literature review, Socio-Econ. Plan. Sci. **46**, 4 (2012).

[12] M. Fingerhuth, T. Babej, and C. Ing, A quantum alternating operator ansatz with hard and soft constraints for lattice protein folding, arXiv:1810.13411.

[13] A. Robert, P. K. Barkoutsos, S. Woerner, and I. Tavernelli, Resource-efficient quantum algorithm for protein folding, npj Quantum Inf. **7**, 38 (2021).

[14] C. Levinthal, Are there pathways for protein folding? J. Chim. Phys. **65**, 44 (1968).

[15] R. Zwanzig, A. Szabo, and B. Bagchi, Levinthal's paradox, Proc. Natl. Acad. Sci. USA **89**, 20 (1992).

[16] S. Piana, J. Klepeis, and D. Shaw, Assessing the accuracy of physical models used in protein-folding simulations: Quantitative evidence from long molecular dynamics simulations, Curr. Opin. Struct. Biol. **24**, 98 (2014).

[17] K. A. Dill, S. B. Ozkan, M. S. Shell, and T. R. Weikl, The protein folding problem, Annu. Rev. Biophys. **37**, 289 (2008).

[18] J. Jumper *et al.*, Highly accurate protein structure prediction with AlphaFold, Nature (London) **596**, 583 (2021).

[19] B. Kuhlman and P. Bradley, Advances in protein structure prediction and design, Nat. Rev. Mol. Cell Biol. **20**, 681 (2019).

[20] M. Dorn, M. B. e Silva, L. S. Buriol, and L. C. Lamb, Three-dimensional protein structure prediction: Methods and computational strategies, Comput. Biol. Chem. **53**, 251 (2014).

[21] S. Kmiecik, D. Gront, M. Kolinski, L. Wieteska, A. E. Dawid, and A. Kolinski, Coarse-grained protein models and their applications, Chem. Rev. **116**, 7898 (2016).

[22] K. Yang, H. Huang, O. Vandans, A. Murali, F. Tian, R. H. Yap, and L. Dai, Applying deep reinforcement learning to the HP model for protein structure prediction, Physica A **609**, 128395 (2023).

[23] O. Van Eck and D. Van Den Berg, Quantifying instance hardness of protein folding within the HP-model, in *2023 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)* (2023), pp. 1–7.

[24] M. Levitt and A. Warshel, Computer simulation of protein folding, Nature (London) **253**, 694 (1975).

[25] D. A. Hinds and M. Levitt, A lattice model for protein structure prediction at low resolution, Proc. Natl. Acad. Sci. USA **89**, 2536 (1992).

[26] K. F. Lau and K. A. Dill, A lattice statistical mechanics model of the conformational and sequence spaces of proteins, Macromolecules **22**, 3986 (1989).

[27] S. Miyazawa and R. L. Jernigan, Residue: Residue potentials with a favorable contact pair term and an unfavorable high packing density term, for simulation and threading, J. Mol. Biol. **256**, 623 (1996).

[28] M. P. Jacobson, G. A. Kaminski, R. A. Friesner, and C. S. Rapp, Force field validation using protein side chain prediction, J. Phys. Chem. B **106**, 11673 (2002).

[29] M. J. Bower, F. E. Cohen, and R. L. Dunbrack, Prediction of protein side-chain rotamers from a backbone-dependent rotamer library: A new homology modeling tool, J. Mol. Biol. **267**, 1268 (1997).

[30] S. Moreno-Hernández and M. Levitt, Comparative modeling and protein-like features of hydrophobic-polar models on a two-dimensional lattice, Proteins **80**, 1683 (2012).

[31] M. Baek *et al.*, Accurate prediction of protein structures and interactions using a three-track neural network, Science **373**, 871 (2021).

[32] A. Perdomo, C. Truncik, I. Tubert-Brohman, G. Rose, and A. Aspuru-Guzik, On the construction of model Hamiltonians for adiabatic quantum computation and its application to finding low energy conformations of lattice protein models, Phys. Rev. A **78**, 012320 (2008).

[33] A. Perdomo-Ortiz, N. Dickson, M. Drew-Brook, G. Rose, and A. Aspuru-Guzik, Finding low-energy conformations of lattice protein models by quantum annealing, Sci. Rep. **2**, 571 (2012).

[34] R. Saito, K. Okuwaki, Y. Mochizuki, R. Nagai, T. Kato, K. Sugisaki, and Y. Minato, Lattice folding simulation of peptide by quantum computation, J. Comput. Chem. Jpn. Int. Ed. **9**, 2022 (2023).

[35] R. Babbush, A. Perdomo-Ortiz, B. O'Gorman, W. Macready, and A. Aspuru-Guzik, Construction of energy functions for lattice heteropolymer models: Efficient encodings for constraint satisfaction programming and quantum annealing, Adv. Chem. Phys. **155**, 201 (2014).

[36] T. Babej, C. Ing, and M. Fingerhuth, Coarse-grained lattice protein folding on a quantum annealer, arXiv:1811.00713.

[37] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, From the quantum approximate optimization algorithm to a quantum alternating operator ansatz, Algorithms **12**, 34 (2019).

[38] C. Outeiral, G. M. Morris, J. Shi, M. Strahm, S. C. Benjamin, and C. M. Deane, Investigating the potential for a limited quantum speedup on protein lattice problems, New J. Phys. **23**, 103030 (2021).

[39] S. Boulebnane, X. Lucas, A. Meyder, S. Adaszewski, and A. Montanaro, Peptide conformational sampling using the quantum approximate optimization algorithm, arXiv:2204.01821.

[40] P. Chandarana, N. N. Hegade, I. Montalban, E. Solano, and X. Chen, Digitized counterdiabatic quantum algorithm for protein folding, Phys. Rev. Appl. **20**, 014024 (2023).

[41] A. Irbäck, L. Knuthson, S. Mohanty, and C. Peterson, Folding lattice proteins with quantum annealing, Phys. Rev. Res. **4**, 043013 (2022).

[42] R. Wong and W.-L. Chang, Fast quantum algorithm for protein structure prediction in hydrophobic-hydrophilic model, J. Parallel Distributed Comput. **164**, 178 (2022).

[43] V. K. Mulligan, H. Melo, H. I. Merritt, S. Slocum, B. D. Weitzner, A. M. Watkins, P. D. Renfrew, C. Pelissier, P. S. Arora, and R. Bonneau, Designing peptides on a quantum computer, bioRxiv (2019), doi: 10.1101/752485.

[44] J. B. Maguire, D. Grattarola, V. K. Mulligan, E. Klyshko, and H. Melo, XENet: Using a new graph convolution to accelerate the timeline for protein design on quantum computers, PLoS Comput. Biol. **17**, e1009037 (2021).

[45] P. A. M. Casares, R. Campos, and M. A. Martin-Delgado, QFold: Quantum walks and deep learning to solve protein folding, Quantum Sci. Technol. **7**, 025013 (2022).

[46] S. Blinov, B. Wu, and C. Monroe, Comparison of cloud-based ion trap and superconducting quantum computer architectures, AVS Quant. Sci. **3**, 033801 (2021).

[47] N. P. D. Sawaya, T. Menke, T. H. Kyaw, S. Johri, A. Aspuru-Guzik, and G. G. Guerreschi, Resource-efficient digital quantum simulation of *d*-level systems for photonic, vibrational, and spin-*s* Hamiltonians, npj Quantum Inf. **6**, 49 (2020).

[48] N. P. Sawaya, A. T. Schmitz, and S. Hadfield, Encoding trade-offs and design toolkits in quantum algorithms for discrete optimization: Coloring, routing, scheduling, and other problems, Quantum **7**, 1111 (2023).

[49] F. Dominguez, J. Unger, M. Traube, B. Mant, C. Ertler, and W. Lechner, Encoding-independent optimization problem formulation for quantum computing, Front. Quantum Sci. Technol. **2** (2023).

[50] T. Tomesh, N. Allen, and Z. Saleem, Quantum-classical trade-offs and multi-controlled quantum gate decompositions in variational algorithms, arXiv:2210.04378.

[51] L. Brocchieri and S. Karlin, Protein length in eukaryotic and prokaryotic proteomes, Nucleic Acids Res. **33**, 3390 (2005).

[52] Y. Wang, S. Zhang, F. Li, Y. Zhou, Y. Zhang, Z. Wang, R. Zhang, J. Zhu, Y. Ren, Y. Tan, C. Qin, Y. Li, X. Li, Y. Chen, and F. Zhu, Therapeutic target database 2020: Enriched resource for facilitating research and early development of targeted therapeutics, Nucleic Acids Res. **48**, D1031 (2020).

[53] B. Berger and T. Leighton, Protein Folding in the Hydrophobic-Hydrophilic (HP) Model is NP-Complete, J. Comput. Biol. **5**, 27 (1998).

[54] W. E. Hart and S. Istrail, Robust proofs of NP-hardness for protein folding: General lattices and energy potentials, J. Comput. Biol. **4**, 1 (1997).

[55] A. S. Fraenkel, Complexity of protein folding, Bull. Math. Biol. **55**, 1199 (1993).

[56] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis, On the complexity of protein folding, J. Comput. Biol. **5**, 423 (1998).

[57] G. A. Barbosa, Quantum half-adder, Phys. Rev. A **73**, 052321 (2006).

[58] N. A. Pierce and E. Winfree, Protein design is NP-hard, Protein Eng., Design Selection **15**, 779 (2002).

[59] R. F. Alford, A. Leaver-Fay, J. R. Jeliazkov, M. J. O'Meara, F. P. DiMaio, H. Park, M. V. Shapovalov, P. D. Renfrew, V. K. Mulligan, K. Kappel, J. W. Labonte, M. S. Pacella, R. Bonneau, P. Bradley, R. L. Dunbrack, R. Das, D. Baker, B. Kuhlman, T. Kortemme, and J. J. Gray, The rosetta all-atom energy function for macromolecular modeling and design, J. Chem. Theory Comput. **13**, 3031 (2017).

[60] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum computation by adiabatic evolution, arXiv:quant-ph/0001106.

[61] M. Müller, K. Hammerer, Y. L. Zhou, C. F. Roos, and P. Zoller, Simulating open quantum systems: From many-body interactions to stabilizer pumping, New J. Phys. **13**, 085007 (2011).

[62] C. W. Warren, J. Fernández-Pendás, S. Ahmed, T. Abad, A. Bengtsson, J. Biznárová, K. Debnath, X. Gu, C. Križan, A. Osman, A. F. Roudsari, P. Delsing, G. Johansson, A. F. Kockum, G. Tancredi, and J. Bylander, Extensive characterization of a family of efficient three-qubit gates at the coherence limit, npj Quantum Inf. **9**, 44 (2023).

[63] I. D. Kivlichan, J. McClean, N. Wiebe, C. Gidney, A. Aspuru-Guzik, G. K.-L. Chan, and R. Babbush, Quantum simulation of electronic structure with linear depth and connectivity, Phys. Rev. Lett. **120**, 110501 (2018).

[64] B. O'Gorman, W. J. Huggins, E. G. Rieffel, and K. B. Whaley, Generalized swap networks for near-term quantum computing, arXiv:1905.05118.

[65] A. Hashim, R. Rines, V. Omole, R. K. Naik, J. M. Kreikebaum, D. I. Santiago, F. T. Chong, I. Siddiqi, and P. Gokhale, Optimized SWAP networks with equivalent circuit averaging for QAOA, Phys. Rev. Res. **4**, 033028 (2022).

[66] S. Kosen *et al.*, Building blocks of a flip-chip integrated superconducting quantum processor, Quantum Sci. Technol. **7**, 035018 (2022).

[67] Z. Cai, C. Luan, L. Ou, H. Tu, Z. Yin, J.-N. Zhang, and K. Kim, Entangling gates for trapped-ion quantum computation and quantum simulation, J. Korean Phys. Soc. **82**, 882 (2023).

[68] IBM quantum development roadmap website (2023), https://www.ibm.com/roadmaps/quantum/.

[69] J. A. Montañez-Barrera, D. Willsch, A. Maldonado-Romo, and K. Michielsen, Unbalanced penalization: A new approach to encode inequality constraints of combinatorial problems for quantum optimization algorithms, Quantum Sci. Technol. **9**, 025022 (2024).

[70] See https://github.com/HannaLinn/resources-qaoa-protein-folding.

[71] C. R. Harris *et al.*, Array programming with NumPy, Nature (London) **585**, 357 (2020).

[72] P. Virtanen *et al.*, SciPy 1.0: Fundamental algorithms for scientific computing in python, Nat. Methods **17**, 261 (2020).

[73] J. D. Hunter, Matplotlib: A 2D graphics environment, Comput. Sci. Eng. **9**, 90 (2007).

[74] X. Gu, J. Fernández-Pendás, P. Vikstål, T. Abad, C. Warren, A. Bengtsson, G. Tancredi, V. Shumeiko, J. Bylander, G. Johansson, and A. F. Kockum, Fast multiqubit gates through simultaneous two-qubit gates, PRX Quantum **2**, 040348 (2021).

[75] K. Mølmer and A. Sørensen, Multiparticle entanglement of hot trapped Ions, Phys. Rev. Lett. **82**, 1835 (1999).

[76] N. Lacroix, C. Hellings, C. K. Andersen, A. Di Paolo, A. Remm, S. Lazar, S. Krinner, G. J. Norris, M. Gabureac, J. Heinsoo, A. Blais, C. Eichler, and A. Wallraff, Improving the performance of deep quantum optimization algorithms with continuous gate sets, PRX Quantum **1**, 020304 (2020).