Journal of Parallel and Distributed Computing 164 (2022) 178–190

Contents lists available at ScienceDirect

Journal of Parallel and Distributed Computing

www.elsevier.com/locate/jpdc

Fast quantum algorithm for protein structure prediction in hydrophobic-hydrophilic model ☆

Renata Wong [a,b,*], Weng-Long Chang [c]

[a] Department of Computer Science and Technology, Nanjing University, Nanjing, PR China
[b] Physics Division, National Center for Theoretical Sciences, Taipei, Taiwan, ROC
[c] Department of Computer Science and Information Engineering, National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan, ROC

ARTICLE INFO

Article history:
Received 24 April 2021
Received in revised form 5 November 2021
Accepted 11 March 2022
Available online 21 March 2022

Keywords:
Molecular algorithms
NP-complete problems
Protein structure prediction
Quantum algorithms

ABSTRACT

In its unfolded form, a protein is a linear sequence of amino acids. Protein structure prediction attempts to find the native conformation for a given protein, which has potential applications in drug and vaccine development. Classically, protein structure prediction is an NP-complete, unsolved computational problem. Quantum computing however promises to improve upon the performance of classical algorithms. Here we develop a quantum algorithm in hydrophobic-hydrophilic model on two-dimensional square lattice to solve the problem for any sequence of length N amino acids with a quadratic speedup over its classical counterpart. This speedup is achieved using Grover's quantum search algorithm. The algorithm can be used for amino acid sequences of arbitrary length. It consists of three stages: (1) preparation of a superposition state that encodes all possible $2^{2(N-1)}$ conformations, (2) calculation of coordinates and energy for each possible conformation in parallel, and (3) finding the conformation with the minimal energy. The asymptotic complexity with regard to space is $O(N^3)$, while the obtained speedup is quadratic compared to the classical counterpart. We have successfully simulated the algorithm on the IBM Quantum's qasm simulator using Qiskit SDK. Also, we have further confirmed the correctness of the results by calculating theoretical probability of finding the right conformation.

1. Introduction

Proteins are linear sequences consisting usually of between 50 and 2000 amino acids [4] that play key roles in most physiological processes of living organisms. The variety of functions essential to sustaining life that proteins perform include, among others, metabolism, immune responses, signal transduction, cell cycle, production of hormones and enzymes. All these functions are made possible by proteins taking on a well-defined structure in the process of folding. This structure is called that protein's *native conformation*. But proteins can misfold, in which case, the misfolding is the root of most endogenous diseases. Understanding the algorithm that makes proteins fold into their native structures is therefore a significant undertaking. Trial-and-error, i.e., testing large numbers of often random compounds for their ability to interfere with a protein's function, is a standard procedure adopted in new drug development; its success rate however is low, and results must be assessed in lengthy clinical trials in later stages [56]. This procedure could be improved if drug tests were performed on *in silico* protein systems before clinical trials take place. For that, an accurate understanding of the folding principles is necessary.

Based on an amino acid sequence, protein structure prediction (PSP) attempts to foresee how the native conformation would look like when folded. By utilizing quantum mechanical properties such as quantum superposition, interference and entanglement, quantum computers promise to solve certain hard problems more efficiently than classical computers. Some of the significant advances in quantum algorithms solving hard problems include Shor's algorithm for integer factorization and discrete logarithms [51], Grover's algorithm for unsorted search [28], quantum algorithm for solving the maximal clique problem [16] or quantum algorithms for linear systems of equations [30,54].

The choice of going quantum is also partly because of the recent successful implementations of quantum algorithms includ-

☆ In this work we demonstrate a quantum algorithm for predicting the tertiary structure of proteins in the hydrophobic-hydrophilic model, which makes use, among others, of superposition, which enables one to process data in parallel. We show that it offers a quadratic improvement in time compared to the classical algorithm and that it requires only polynomial space with respect to the number of amino acids. Furthermore, we also test the correctness of the algorithm on IBM qasm simulator using the Qiskit SDK.

* Corresponding author at: Physics Division, National Center for Theoretical Sciences, Taipei, Taiwan, ROC.
E-mail addresses: renata.wong@protonmail.com (R. Wong), changwl@cc.kuas.edu.tw (W.-L. Chang).

https://doi.org/10.1016/j.jpdc.2022.03.011
0743-7315/

ing Shor's factoring algorithm by various techniques [36,43,40], Grover's search algorithm [50,25], as well as boson sampling [2, 53,37], among others.

Here we solve the PSP problem for the hydrophobic-hydrophilic model (HP) [21] in two dimensions on a square lattice, which is one of the most widely studied methods for PSP [57,60,41]. PSP in the HP model falls within the complexity class of NP-complete problems [19].

In principle, NP-complete problems can be solved by oracular search algorithms. Such algorithms incorporate an oracle, which is also often referred to as a black-box. The first such search algorithm was developed by Grover [29]. Some improvements then followed (e.g., [49]). It has been proven that oracular quantum algorithms can provide up to a quadratic speedup over classical algorithms for the same problem [28,11]. Therefore, an optimal oracular quantum algorithm for any NP-complete problem, including any model of PSP, should work in asymptotic time $O(\sqrt{2^N})$, where N is the length of amino acid sequence. By using Grover's algorithm to find the conformation with the minimal energy, our algorithm achieves in the worst case $O(N^3\sqrt{2^N})$ for time complexity and is therefore optimal. The worst-case time performance of a classical algorithm for the problem is not known as it is not possible to uniquely represent all the possible conformations on any state-of-the-art classical supercomputer (these store currently an order of 200 PB). Consider a smallest protein of 50 amino acids in length. It has 2^{98} possible conformations on a square lattice. As largest classical memories provide only several hundred PB, clearly, this number cannot be stored classically. On the other hand, quantum computers would only require polynomial space ($O(N^3)$) for the task.

The currently available quantum computational devices offer mostly 14-15 qubits, while devices of up to 79 qubits are under intensive development [18]. Representing all conformations of a smallest protein of 50 amino acids in length would require 98 qubits. Even if this number of qubits were available, additional qubits would be needed to calculate the lattice coordinates and energies for each conformation, and to find the conformation with the lowest energy. Therefore, it is not possible presently to simulate our algorithm on any real proteins. The largest sequence that can be simulated has length $N = 3$.

Besides the limited availability of qubits, one also has to consider the fact that the currently existing quantum devices are not fault-tolerant due to decoherence of physical systems. Increasing the number of qubits and gates in a circuit makes the computation intractable [18], effectively rendering any larger circuits impossible to provide a desired outcome upon measurement. As an intermediate solution, quantum simulators facilitate a reliable testing in an idealized quantum environment. For example, IBM qasm simulator allows for simulations of up to 32 qubits. Executing an instance of our algorithm for sequence length $N = 3$ on this simulator requires 25 qubits at minimum and the algorithm has to be tailored to this length. As outlined in Section 5, a general case of experimental validation for $N = 3$ would involve $16N^3 - 24N^2 - 12N + 63 = 243$ qubits. Due to resource restrictions, our validation had to be simplified to such an extent that the number of qubits did not exceed 32. Such a simplification was not possible for $N \geq 4$.

A qubit count of 50 and above is considered transcending the classical realm and would imply "quantum supremacy" [12]. Simulations of quantum systems of 50 qubits and above take around 16 PB of RAM [17] on a supercomputer and are limited in circuit depth. Google and IBM proposed efficient methods to simulate low-depth circuits of above 49 qubits. IBM successfully simulated a depth of 27 in 2017 on the Vulcan supercomputer at Lawrence Livermore National Laboratory [42]. Google [13] found that circuits with 7×7 qubits and depth 40 remain out of reach. Some extensions of simulation techniques have been presented that are to allow for simulation of quantum circuits with greater depth. For example, 7×7 circuits can in principle be fully simulated with all amplitudes calculated to arbitrary depth in less than a day for a depth-83 circuit by leveraging secondary storage [42]. These extensions are theoretical and yet to be tested computationally [42]. For the case of $N = 3$, our algorithm requires between $3,272 - 3,296$ (Table 2), which corresponds closely to the depth of the circuits. Due to this depth, it is currently impossible to test our algorithm for more than $N = 3$.

In order to execute these circuits on a quantum device, they must be transpiled first. In general, transpilation involves a decomposition of a circuit into single-qubit gates and CNOT gates only. For the case of $N = 3$, the required number of non-transpiled gates is between $3,272 - 3,296$ (Table 2). To the best of our knowledge, the currently most resource efficient Clifford+T decomposition method for the Toffoli gate involves its decomposition into 6 CNOTS, 7 T gates and 2 Hadamard gates [58]. Under this decomposition, the depth of our circuits will range between $13,296 - 13,348$. Assuming a CNOT gate fidelity of 99.77% [35], the probability of success in measuring the correct outcome will be $0.9977^{4296} = 0.005\%$, where the exponent is the number of CNOT gates in the transpiled circuit. Clearly, an execution on a quantum device, even if such were available, would be inconclusive due to the cumulative error effect. With the rapid development in the fidelity of quantum devices though, we hope that the algorithm could be executed on real proteins in not-so-distant future.

This paper is an interdisciplinary study between quantum computing and bioinformatics, and is organized as follows. Section 2 discusses other quantum approaches to protein structure prediction. Section 3 introduces the widely studied hydrophobic-hydrophilic model for protein structure prediction. The proposed quantum algorithm, which is set in the hydrophobic-hydrophilic model, is then described in detail in Section 4. Section 5 gives the complexity assessment for the algorithm. Section 6 presents the two experiments we have conducted on the IBM quantum simulator. Section 7 lists the resources used in the simulations. Section 8 informs on source code availability, while Section 9 concludes. Appendix A provides additional corroboration of the experimental results obtained in Section 6 by calculating the theoretical probability of finding the solution.

Parallelization in quantum computing In analogy to classical parallel computation, quantum computing can be seen as a type of parallelization. A qubit is a two-level system of the form:

$$|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1}$$

where the coefficients α and β are referred to as probability amplitudes, which are complex numbers. The states $|0\rangle$ and $|1\rangle$ stand for the two possible values that the state $|\phi\rangle$ attains upon measurement. A qubit is the smallest quantum state. A quantum state is probabilistic with the modulus squared of each probability amplitude indicating the probability of the component state being observed upon measurement. The probabilities of all component states must therefore add to 1.

Parallelization in quantum computing is achieved through quantum superposition, in which a total of 2^k unique paths are generated for k qubits:

$$|\psi\rangle = \sum_{i=0}^{2^k-1} \alpha_i\,|i\rangle \tag{2}$$

Here, α_i is the probability amplitude of each path $|i\rangle$. Initially, the probability amplitude of each path in a superposition is usually equal. After initialization, quantum state is operated on by applying various quantum gates, which operation can be described in

terms of unitary operators acting on vectors. The purpose of this is to reduce as much as possible the state components in superposition. This is achieved by constructive and destructive interference. Both types of interference manipulate the probability amplitudes of component basis states in a quantum state, where the complex amplitudes can be either positive or negative.

Despite the quantum state remaining in a superposition for the entire computation time, upon measurement, the output of a quantum computation is a single state (path). This is described by the measurement postulate of quantum mechanics, which also states that the output of a measurement is inherently probabilistic. The particular output state (path $|i\rangle$) is observed with probability given by the modulus squared of the respective probability amplitude.

Highlights and significance of this work The following are the main contributions of the paper:

- In this work, we show how to implement our algorithm for amino acid sequences of arbitrary length down to single quantum gates. As such it can be readily put into a desired architecture and tested.
- Consisting of only the eigenstates $|0\rangle$ and $|1\rangle$ of Pauli-Z operator, all the input states of our algorithm can thus be readily physically encoded into qubits, which is not always possible in quantum algorithms.
- With the PSP problem being NP-complete even for the simplest classical models, our quantum algorithm provides a quadratic speedup over its classical counterparts. The space complexity is reduced to mere N^3 thanks to the use of quantum superposition, while classically it is exponential for the model.
- It is a known problem in quantum computing that an oracle is often very difficult to specify and to physically encode into qubits. Our quantum algorithm achieves both. Namely, we provide a clear instruction for the structure of the oracle to be used in this algorithm and the oracle is straightforward to implement as it consists of only the eigenstates of the Pauli-Z operator.
- We experimentally verify the correctness of the algorithm by executing it on IBM's quantum simulator. The number of the quantum gates used in the verification is some of the largest tested up to date.
- We confirm the correctness of the verification results by calculating the theoretical probability of success. With this, we have confirmed that the algorithm will indeed provide the solution to the PSP problem as defined within the confines of the HP model.

2. Related works

In recent years, there have been developments in experimental protein structure prediction using quantum mechanical principles, many of them using quantum adiabatic computation (or a related concept of quantum annealing), a model different from the circuit model used in the present work. Here, we briefly summarize some of the results.

Quantum adiabatic computation [23] designs its calculations based on the adiabatic theorem [14]. It has been demonstrated to be polynomially equivalent to quantum circuit model [3]. Quantum adiabatic computation model assumes the qubits to be coherent. Quantum annealing is a version of quantum adiabatic optimization in which the qubits are strongly coupled to their environment throughout, but still maintain *some* quantum coherence [1]. This is in contrast to quantum circuit model where high qubit coherence and good isolation from environment are basic assumptions that affect gate fidelity and computational error rates. As pointed out in Section 1, these error rates limit the performance of present day quantum algorithms. The lack of strong coherence requirement also implies that quantum annealing methods can avail themselves with a relatively large amount of qubits as they do not have to uphold coherence constraints. The currently largest number factorized using Shor's quantum integer factorization algorithm [52] is 21 and was achieved in 2012 [38]. Due to accumulation of errors it was not possible to factorize the number 35, attempted in 2019 [6]. Much larger numbers have however been factored using quantum annealing. For example, Jiang et al. [34] factorized 15, 143, 59989, and 376289 using 4, 12, 59, and 94 logical qubits respectively.

Quantum annealing is at heart in the experiments conducted by Perdomo-Ortiz et al. [45] which looked for low-energy conformations in lattice models. 8 different experiments of up to 81 superconducting bits were carried out in the study. Instead of the hydrophobic-hydrophobic interactions used in our algorithm, the authors considered a more complex Miyazawa-Jernigan [39] interactions among amino acids in an amino acid sequence of length 6. Among the reported results, an 8-qubit experiment predicted the probability of measuring the ground state as 80.7%. Yet, as the authors point out, the odds of measuring the ground state are not necessarily high, as exemplified on an 81 qubit experiment in the same study in which only 13 out of 10,000 measurements yielded the desired solution. Another work using quantum annealing is that of Babej and Ing [9], involving an implementation of Chignolin, a sequence of 10 amino acids, on a planar lattice and of Trp-Cage (8 amino acids) on a cubic lattice using the D-Wave 2000Q quantum processing unit with 2048 superconducting qubits. Out of the 102,400,000 samples collected from the QPU in the 10-amino acid experiment, 24,950 samples were found to be the correct ground state. This implies a success probability of 0.000244. For the Trp-Cage fragment a total of 204,800,000 samples were collected and 4,957 of them were the correct lattice fold. This gives a success probability of 2.42×10^{-5}. Babbush et al. [8] give a good overview of how to construct energy functions for protein structure prediction for the quantum annealing regime.

Regarding models other than quantum annealing, we describe briefly three works. Perdomo-Ortiz et al. [44] implemented an algorithm for protein structure prediction using a 2-dimensional square lattice in the hydrophobic-hydrophilic model in the quantum adiabatic regime. The authors were able to solve in detail the four amino acid sequence HPPH. Fingerhuth et al. [26] used quantum variational algorithm to tackle the lattice protein folding problem using Quantum Approximate Optimization Algorithm (QAOA) [24]. The work considered a 4 amino acid sequence PSVK on both planar and cubic lattices. QAOA is a hybrid approach where the computation is done iteratively consisting of an evaluation of the output of a quantum circuit by classical optimizers and a subsequent execution of the quantum circuit with the optimized values. For the planar lattice, the probability of ground state varied depending on circuit depth and used mixer Hamiltonian. The median ground state probabilities ranged between 0.055 for the XY_{simple} mixer Hamiltonian and 0.019 for the $XZ_{overlap}$ Hamiltonian. The highest value reported was 0.477 (XY_{simple}). Robert et al. [48] used a combination of variational quantum algorithms specifically adapted to classical cost functions and evolutionary strategies to simulate the folding of the 10 amino acid Angiotensin on 22 qubits using the Miyazawa-Jernigan interaction model. Optimization was done using CVaR-VQE [10], a variation of Variational Quantum Eigensolver [46] called Conditional Value-at-Risk (CVaR). As 22 qubits are too large for encoding in state-of-the-art quantum hardware, simulations were carried out using a realistic parametrization of the noise and the results were obtained using 128 and 1024 measurements. The total probabilities of find-

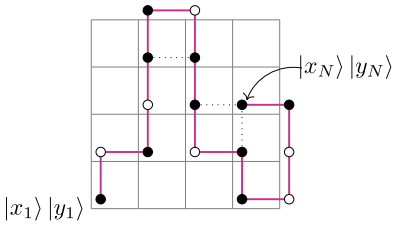

Fig. 1. A possible conformation embedded into a square lattice with hydrophobic (•) and hydrophilic (∘) amino acids and hydrophobic lattice contacts (dotted) indicated. $|x_1\rangle\,|y_1\rangle$ and $|x_N\rangle\,|y_N\rangle$ are the respective coordinates of the first and the last amino acid in the sequence.

ing low-energy conformations (energy below 0) added up to 89.5% (small sampling) and 100% (large sampling). The study also implemented a 7 amino acid sequence on a quantum hardware using 9 qubits. The averaged ground state probability for this sequence was above 20% with the best probability being 33%.

3. The hydrophobic-hydrophilic model

Let $a = a_1 a_2 \ldots a_N$ denote a sequence consisting of N amino acids that encode a protein. Each a_e denotes the amino acid located at a position $1 \le e \le N$ in the sequence. Under the hydrophobic-hydrophilic (HP) model for protein structure prediction, each amino acid is classified either as hydrophobic (H) or hydrophilic (P). This classification is based on experimentally obtained parameters [59]. The information of whether an amino acid is hydrophobic or hydrophilic can be stored in a single qubit. One of the simplest encodings would be $|1\rangle$ for a hydrophobic amino acid and $|0\rangle$ for a hydrophilic amino acid.

Under the two-dimensional HP-model, the amino acid sequence is mapped into a square lattice. An example mapping is shown in Fig. 1. Finding the native conformation for a protein is usually subdivided into three steps. In the first step, the number of all possible conformations is determined. In the second step, the energy values for each possible conformation are calculated based on a scoring function that is particular to the given PSP model. And in the last step, the conformation with the lowest free energy is selected.

Under the HP model, a major contribution to free energy is due to interactions between hydrophobic amino acids, which form a core within the protein. Each conformation is given a score according to the scoring function

$$V = -\left|\{(|a_i\rangle, |a_j\rangle)\}\right| \tag{3}$$

where a_i, a_j $(1 \le i, j \le N)$ are hydrophobic amino acids that are adjacent in the lattice while at the same time not adjacent in the amino acid sequence. Hence, function V corresponds to the magnitude of the set containing all pairs of hydrophobic amino acids that are adjacent in the lattice but not in the sequence. This gives the number of all contacts between a_i and a_j in the lattice for a conformation. The minus sign signifies that the highest number of such contacts corresponds to the lowest energy value. Three of such contacts are shown in Fig. 1, represented by dotted lines. These contacts are often referred to as loose contacts. The number of loose contacts is inversely proportional to the amount of free energy for a conformation, i.e., the higher the number of loose contacts the lower the free energy.

On the choice of lattice Square and cubic lattices are the most widely studied lattice types for the planar and 3D HP model respectively. However, other types of shapes have been proposed with various reasons in mind. One of those reasons is the parity problem, which affects both square and cubic lattices, and in which residues of the same parity cannot make hydrophobic contact [22]. To counteract this problem, it was suggested to use triangular lattices or square lattices with diagonals. As triangular lattices may introduce sharp turns in adjacent amino acids, hexagonal lattices were proposed to mitigate this issue [33]. Another reason for choosing a particular lattice type could be for example to better account for the plausibility of angles, which made Robert et al. [48] choose the tetrahedral lattice for their study. Besides the square lattice being one of the most widely used lattice shape for the HP model to study protein folding [8], another reason for choosing the square lattice for our study was the availability of resources. In order to guarantee that the algorithm could be implemented experimentally so as to prove its correctness we were bound to make the implementation as simple as possible yet as accurate as possible. The square lattice seemed to combine the two aspects well.

4. The algorithm

In order to implement a mapping from a sequence into a lattice, the structure of the lattice has to be accounted for. In this paper, we consider the square lattice, which is the standard lattice used with the two-dimensional HP model. It has four possible directional transitions out of a node. To represent transitions from one amino acid to another in this lattice, absolute directions are used: N (northward), W (westward), S (southward), and E (eastward). The transitions are defined in terms of the usual Cartesian coordinates: $N(x, y) = (x, y + 1)$, $E(x, y) = (x + 1, y)$, $S(x, y) = (x, y - 1)$, $W(x, y) = (x - 1, y)$. For example, the expression $N(x, y) = (x, y + 1)$ indicates that a transition northwards from the given position will involve increasing the y coordinate by 1 to reflect the fact that the new coordinate is that of an amino acid (or lattice site) located north of the given coordinate.

In the following subsections, we describe each of the algorithm's steps.

4.1. Initialization

Superscripts are used throughout to indicate a specific binary value of a state, if any. For a sequence $|a\rangle = |a_1 a_2 \cdots a_N\rangle$ of amino acids, $\left|a_e^1\right\rangle$ indicates that the corresponding amino acid a_e is hydrophobic (value 1), while $\left|a_e^0\right\rangle$ indicates that it is hydrophilic (value 0). The sequence $|a\rangle$ is stored in a quantum register of length N:

$$|a\rangle = \bigotimes_e |a_e\rangle\,, \quad 1 \le e \le N \tag{4}$$

where $\otimes$ is the tensor product.

Each amino acid on a lattice is identified by its x and y coordinates:

$$|x\rangle = \bigotimes_e \left|x_e^0\right\rangle \quad , \quad |y\rangle = \bigotimes_e \left|y_e^0\right\rangle \tag{5}$$

We assume that the coordinates of the first amino acid a_1 are fixed at the origin $(0, 0)$ of the lattice. With this, for a sequence of length N representing all conformations will require a total of $2N - 1$ values for x and the same amount for y. The conformations will be generated in superposition. These are: the value 0, the four positive values $1, 2, \ldots, N - 1$, and the four negative values $-1, -2, \ldots, -(N - 1)$. Based on this number $(2N - 1)$, the number of qubits necessary to uniquely represent each coordinate in binary is $t = \lceil \log_2(2N - 1) \rceil$.

Two binary variables $\left|w_{d,p=1}\right\rangle$ and $\left|w_{d,p=2}\right\rangle$ are used to indicate the transition direction in the lattice from $(d - 1)$-th site to d-th site for $2 \le d \le N$. As the first amino acid in a conformation can

Table 1
Transition directions between adjacent amino acids.

$\|w_{d,2}\rangle$	$\|w_{d,1}\rangle$	from $(d-1)$-th site to d-th site
0	0	northward (N)
0	1	eastward (E)
1	0	southward (S)
1	1	westward (W)

be assumed to have a fixed position, there are $N-1$ transitions for each conformation. Therefore, $N-1$ quantum registers of length 2 are needed to encode these two variables for the entire amino acid sequence. Their initial states are all set to 0:

$$|w\rangle = \bigotimes_d \bigotimes_p \left|w^0_{d,p}\right\rangle \tag{6}$$

where $2 \le d \le N$ and $1 \le p \le 2$. The four possible transition directions are encoded as provided by Table 1.

Coordinate transitions will require both addition and subtraction. In order to simplify this operation, we adopt the two's complement notation. This notation encodes the information that a number is negative in the number itself, thereby not differentiating between subtraction and addition. This way, only quantum addition, but not subtraction must be implemented for coordinate transitions. The two's complement of 1 is

$$|\alpha\rangle = |0\rangle^{\otimes t-1}|1\rangle \tag{7}$$

while the two's complement of -1 is

$$\left|\alpha^*\right\rangle = |1\rangle^{\otimes t} \tag{8}$$

where t is the number of qubits necessary to represent a coordinate.

After the system has been initialized, we proceed by setting it into a superposition state over vector $|w\rangle$. This vector uniquely identifies each of the $2^{2(N-1)}$ possible conformations of a sequence of length N. The superposition state:

$$\frac{1}{\sqrt{2^{2(N-1)}}} \sum_{w=0}^{2^{2(N-1)}-1} |w\rangle \tag{9}$$

is obtained by applying $2(N-1)$ Hadamard gates to the vector $|w\rangle$. We have omitted here all other quantum registers, which are not affected by this operation. With this, the conformational space of a protein can be calculated. This space encompasses all candidates for a native conformation.

4.2. Constructing conformational space

Let $|x_d\rangle$ and $|y_d\rangle$ denote the coordinates of the d-th amino acid in a sequence for $2 \le d \le N$ (the first amino acid has fixed coordinates and can be omitted here). The conformational space is constructed in two steps as shown in the algorithm in Fig. 2.

First, in rows 2-3 of the algorithm, CNOT gates are applied. The gates are indicated by the symbol $\oplus$ and correspond to addition modulo 2. Each of the gates copies coordinates $\left|x_{d-1}\right\rangle$ and $\left|y_{d-1}\right\rangle$ of the $(d-1)$-th amino acid to $|x_d\rangle$ and $|y_d\rangle$ of the d-th amino acid, respectively, for $2 \le d \le N$. Then, the coordinates of the next neighbor $\left|a_{d+1}\right\rangle$ for each amino acid $|a_d\rangle$ in a conformation are calculated in rows 4 through 11 depending on the value of vector $\left|w_{d,2}w_{d,1}\right\rangle$. Adding a 1 ($|\alpha\rangle$) increases the coordinate value by 1, while adding the two's complement of 1 ($|\alpha^*\rangle$) decreases the value by 1. For example, if $\left|w_{d,2}w_{d,1}\right\rangle = |00\rangle$, the coordinates of the site $\left|a_{d+1}\right\rangle$ north of $|a_d\rangle$ are calculated by adding $|\alpha\rangle$ to the

```
1:  for d ← 2 to N do
2:    |x_d⟩ ← |x_{d−1} ⊕ x^0_d⟩
3:    |y_d⟩ ← |y_{d−1} ⊕ y^0_d⟩

4:    if |w_{d,2}w_{d,1}⟩ = |00⟩ then
5:      |y_d⟩ ← |y_d⟩ + |α⟩
6:    else if |w_{d,2}w_{d,1}⟩ = |01⟩ then
7:      |x_d⟩ ← |x_d⟩ + |α⟩
8:    else if |w_{d,2}w_{d,1}⟩ = |10⟩ then
9:      |y_d⟩ ← |y_d⟩ + |α*⟩
10:   else if |w_{d,2}w_{d,1}⟩ = |11⟩ then
11:     |x_d⟩ ← |x_d⟩ + |α*⟩
12:   end if
13: end for
```

Fig. 2. Algorithm for constructing conformational space.

$|y\rangle$ coordinate of $|a_d\rangle$. These coordinates are stored in the $|y\rangle$ coordinate for site $\left|a_{d+1}\right\rangle$. If $\left|w_{d,2}w_{d,1}\right\rangle = |10\rangle$, the coordinates of the southern neighbor are calculated by adding $|\alpha^*\rangle$ to $|y_d\rangle$, and we obtain $\left|y_{d+1}\right\rangle$. This step employs a quantum adder. In our case, we make use of the quantum ripple-carry adder [20] by amending it with two control qubits that control which operations, depending on the value $\left|w_{d,2}w_{d,1}\right\rangle$, take place.

Upon completing the operations in Fig. 2, each conformation has its unique set of x and y coordinates, corresponding to each amino acid in that conformation, stored in the superposition state.

4.3. Energy calculation

The energy depends on the number of loose contacts that a given conformation possesses. Therefore, the energy can be calculated from the coordinates of amino acids. For each loose contact present, a value $|1\rangle$ will be stored in the following vector:

$$|\Phi\rangle = \bigotimes_k \bigotimes_j \bigotimes_p \left|\Phi^0_{k,j,p}\right\rangle \tag{10}$$

where $1 \le k, j \le N$, and $0 \le p \le 3$. The combination of the three indices provides the information about the presence ($|1\rangle$) or absence ($|0\rangle$) of a loose contact. For example, $\left|\Phi_{2,5,1}\right\rangle = |1\rangle$ indicates that amino acid $k=2$ is hydrophobic and forms a loose contact with amino acid $j=5$, which is also hydrophobic. Furthermore, j is the eastern neighbor of k in the lattice. This last information is specified by index p, which we define as $p = w_{d,2}w_{d,1}$ (see Table 1). In this case, $p=1$, or 01 in binary, which corresponds to the eastern neighbor in Table 1. If, on the other hand, $\left|\Phi_{2,5,1}\right\rangle = |0\rangle$, then at least one of these conditions is not fulfilled, e.g., (1) amino acid $k=2$ is not hydrophobic, or (2) amino acid $j=5$ is not hydrophobic, or (3) j and k are not adjacent in the lattice. Hence, no loose contact is present for this combination of the three indices.

Vector $|\Psi\rangle$ is calculated by the algorithm in Fig. 3. The algorithm makes use of auxiliary quantum registers. $8N^2$ registers, contained in the states $|x^*\rangle$ or $|y^*\rangle$, will encode the x and y coordinates of each k's four neighbor sites. Each register is of size t qubits. As before, $1 \le k, j \le N$, $0 \le p \le 3$:

$$\left|x^*\right\rangle = \bigotimes_k \bigotimes_j \bigotimes_p \left|x^{*0}_{k,j,p}\right\rangle \tag{11}$$

$$\left|y^*\right\rangle = \bigotimes_k \bigotimes_j \bigotimes_p \left|y^{*0}_{k,j,p}\right\rangle \tag{12}$$

All coordinates are initialized to 0.

Further, a state $|R\rangle$ will store the result of comparing coordinates of the k-th amino acid with those of the j-th amino acid:

1: **for** $k \leftarrow 1$ to $N-3$ **do**
2: **for** $j \leftarrow k+3$ to N **do**
3: **for** $p \leftarrow 0$ to 3 **do**
4: $\left|x^*_{k,j,p}\right\rangle \leftarrow \left|x_k \oplus x^{*0}_{k,j,p}\right\rangle$
5: $\left|y^*_{k,j,p}\right\rangle \leftarrow \left|y_k \oplus y^{*0}_{k,j,p}\right\rangle$
6: **if** $p=0$ **then** $\left|y^*_{k,j,p}\right\rangle \leftarrow \left|y^*_{k,j,p}\right\rangle + |\alpha\rangle$
7: **else if** $p=1$ **then** $\left|x^*_{k,j,p}\right\rangle \leftarrow \left|x^*_{k,j,p}\right\rangle + |\alpha\rangle$
8: **else if** $p=2$ **then** $\left|y^*_{k,j,p}\right\rangle \leftarrow \left|y^*_{k,j,p}\right\rangle + |\alpha^*\rangle$
9: **else if** $p=3$ **then** $\left|x^*_{k,j,p}\right\rangle \leftarrow \left|x^*_{k,j,p}\right\rangle + |\alpha^*\rangle$
10: **end if**
11: $\left|x^*_{k,j,p}\right\rangle \leftarrow \left|x_j \oplus x^*_{k,j,p}\right\rangle$
12: $\left|y^*_{k,j,p}\right\rangle \leftarrow \left|y_j \oplus y^*_{k,j,p}\right\rangle$
13: $|R_{k,j,p,1}\rangle \leftarrow \left|\left(\overline{y^*_{k,j,p,1}} \wedge R^1_{k,j,p,0}\right) \oplus R^0_{k,j,p,1}\right\rangle$
14: **for** $h \leftarrow t$ to 2 **do**
15: $|R_{k,j,p,h}\rangle \leftarrow \left|\left(\overline{y^*_{k,j,p,h}} \wedge R_{k,j,p,h-1}\right) \oplus R^0_{k,j,p,h}\right\rangle$
16: **end for**
17: **for** $h \leftarrow 2t$ to $t+1$ **do**
18: $|R_{k,j,p,h}\rangle \leftarrow \left|\left(\overline{x^*_{k,j,p,h-t}} \wedge R_{k,j,p,h-1}\right) \oplus R^0_{k,j,p,h}\right\rangle$
19: **end for**
20: $|\delta_{k,j,p,1}\rangle \leftarrow \left|(a_k \wedge R_{k,j,p,2t}) \oplus \delta^0_{k,j,p,1}\right\rangle$
21: $|\delta_{k,j,p,2}\rangle \leftarrow \left|(a_j \wedge \delta_{k,j,p,1}) \oplus \delta^0_{k,j,p,2}\right\rangle$
22: $|\Psi_{k,j,p}\rangle \leftarrow \left|\delta_{k,j,p,2} \oplus \Psi^0_{k,j,p}\right\rangle$
23: **end for**
24: **end for**
25: **end for**

Fig. 3. Algorithm for calculating energy.

$$|R\rangle = \bigotimes_k \bigotimes_j \bigotimes_p \bigotimes_h \left|R^0_{k,j,p,h}\right\rangle \otimes \left|R^1_{k,j,p,0}\right\rangle \tag{13}$$

where $1 \le k, j \le N$, $0 \le p \le 3$, and $1 \le h \le 2t$. And finally, the state

$$|\delta\rangle = \bigotimes_k \bigotimes_j \bigotimes_p \bigotimes_u \left|\delta^0_{k,j,p,u}\right\rangle \tag{14}$$

is used to store the intermediate result of checking whether the k-th amino acid $|a_k\rangle$ and the j-th amino acid $\left|a_j\right\rangle$ are adjacent in a given conformation. The indices range as follows: $1 \le k, j \le N$, $0 \le p \le 3$, and $1 \le u \le 2$.

To find the energy values associated with each conformation, the coordinates of every amino acid k must be compared with the coordinates of neighboring lattice sites j to check whether the two neighbors are both hydrophobic and are adjacent in the lattice but not in the sequence. To that end, the coordinates of every amino acid are compared with the coordinates of all other amino acids in a conformation with the exception of the second and the third relative to the considered amino acid. Since the second amino acid (in relative terms) is directly connected to the first in the sequence, their comparison can be omitted as they will never fulfill the condition of not being adjacent in the sequence. Furthermore, due to the structure of the square lattice, the third amino acid (in relative terms) cannot be adjacent to the first in the lattice. Therefore, the comparison will start from the fourth amino acid (in relative terms). This reduces the number of comparisons of sites k and j to $1 \le k \le N-3$ and $k+3 \le j \le N$ as specified in the algorithm in Fig. 3. Parameter $0 \le p \le 3$ indicates the respective neighbor j of k as specified in Table 1. $p=0$ indicates the northern neighbor, $p=1$ the eastern neighbor, $p=2$ the southern neighbor, while $p=3$ indicates the western neighbor.

Finding the energy for each conformation comprises four steps. In the first step, operations in rows 4-5 of the algorithm are executed. These operations correspond to copying coordinates $|x_k\rangle$ and $|y_k\rangle$ of the k-th amino acid to all the coordinates $\left|x^*_{k,j,p}\right\rangle$ and $\left|y^*_{k,j,p}\right\rangle$, respectively. In the second step, as shown in rows 6 through 9 of the algorithm, the coordinates $\left|x^*_{k,j,p}\right\rangle$ and $\left|y^*_{k,j,p}\right\rangle$ are adjusted to store the respective potential neighbors of k depending on the value of p. This step constructs all four possible neighbors j for every amino acid k in a conformation. In the third step in rows 11-12 of the algorithm, the CNOT gate checks whether amino acid j is one of the four neighbors of k. This information is stored in $\left|x^*_{k,j,p}\right\rangle$ and $\left|y^*_{k,j,p}\right\rangle$. j is an actual neighbor of k only when the CNOT gate results in $\left|x^*_{k,j,p}\right\rangle = |0\rangle^{\otimes t}$ and $\left|y^*_{k,j,p}\right\rangle = |0\rangle^{\otimes t}$. CNOT corresponds to the exclusive OR operation. It compares the state $\left|x_j\right\rangle$ with $\left|x^*_{k,j,p}\right\rangle$, and $\left|y_j\right\rangle$ with $\left|y^*_{k,j,p}\right\rangle$. A 0 is output only when the two strings are identical.

In the fourth step, operations in rows 13 through 22 are executed. Together, these operations determine whether k and j are both hydrophobic and are adjacent in the lattice but not in the sequence. As $\left|R_{k,j,p,0}\right\rangle = |1\rangle$ always, as long as both coordinates $|x^*\rangle$ and $|y^*\rangle$ are of the form $|0\rangle^{\otimes t}|0\rangle^{\otimes t}$ after executing the third step, the value 1 will be propagated all the way down from row 13 through 19. Then, in rows 20 and 21, the hydrophobicity status of both the k-th amino acid $|a_k\rangle$ and the j-th amino acid $\left|a_j\right\rangle$ is checked. If $|a_k\rangle$ is hydrophobic, then $|a_k\rangle = |1\rangle$ and therefore $\left|\delta_{k,j,p,1}\right\rangle = |1\rangle$ if and only if $\left|R_{k,j,p,2t}\right\rangle = |1\rangle$. If $\left|a_j\right\rangle$ is hydrophobic, then $\left|a_j\right\rangle = |1\rangle$ and therefore $\left|\delta_{k,j,p,2}\right\rangle = |1\rangle$, given that $\left|\delta_{k,j,p,1}\right\rangle = |1\rangle$.

The final result is recorded in the vector $\left|\Psi_{k,j,p}\right\rangle$ in row 22. $\left|\Psi_{k,j,p}\right\rangle = |1\rangle$ if and only if all three conditions are fulfilled, that is if both $|a_k\rangle$ and $\left|a_j\right\rangle$ are hydrophobic and are adjacent in the lattice while being not adjacent in the sequence. Otherwise, $\left|\Psi_{k,j,p}\right\rangle = |0\rangle$. As an example, $\left|\Psi_{1,4,2}\right\rangle = |1\rangle$ indicates that $k=1$-st amino acid in the sequence has $j=4$-th amino acid in the sequence as its southern neighbor ($p=2$) and that both $|a_k\rangle$ and $\left|a_j\right\rangle$ are hydrophobic.

4.4. Summing up energies

After completing the operations in the previous section, the adjacency information is stored in the state $|\Phi\rangle$. These piecemeal values must be added up for each conformation in order to obtain a string encoding their sum. The state that contains all such strings in superposition is

$$\bigotimes_k \bigotimes_p \bigotimes_j \left|z_{k,p,j}\right\rangle \tag{15}$$

where $1 \le k \le N-3$, $0 \le p \le 3$, and $0 \le j \le 4(k-1)+p$.

In order to calculate this sum, auxiliary qubits $\left|z_{k,p,j}\right\rangle$ and $\left|z_{k,p,j+1}\right\rangle$ are needed, as shown in the algorithm in Fig. 4. The number of values of parameter j is $n = 4(k-1)+p+1$. By substituting the highest value of k and p into n, one can see that $n = 4(N-4)+4$. With this, the total number of additional qubits is calculated to be the partial sum

$$\sum_{m=1}^{n} m = \frac{n(n+1)}{2} = \frac{(4N-12)(4N-11)}{2} \tag{16}$$

All the $\left|z_{k,p,j}\right\rangle$ and $\left|z_{k,p,j+1}\right\rangle$ states in the algorithm are initialized to 0 with the exception of $\left|z_{0,3,0}\right\rangle$, which is initialized to 1. For

for $k \leftarrow 1$ to $N-3$ **do**
  for $p \leftarrow 0$ to 3 **do**
    for $j \leftarrow 4(k-1)+p$ to 0 **do**
      if $p = 0$ **then**
        $|z_{k,p,j+1}\rangle \leftarrow \left(\bigvee_{i=k+3}^{N} |\Psi_{k,i,p}\rangle \wedge |z_{k-1,3,j}\rangle\right) \oplus |z_{k,p,j+1}\rangle$
        $|z_{k,p,j}\rangle \leftarrow \left(\overline{\bigvee_{i=k+3}^{N} |\Psi_{k,i,p}\rangle} \wedge |z_{k-1,3,j}\rangle\right) \oplus |z_{k,p,j}\rangle$
      else
        $|z_{k,p,j+1}\rangle \leftarrow \left(\bigvee_{i=k+3}^{N} |\Psi_{k,i,p}\rangle \wedge |z_{k,p-1,j}\rangle\right) \oplus |z_{k,p,j+1}\rangle$
        $|z_{k,p,j}\rangle \leftarrow \left(\overline{\bigvee_{i=k+3}^{N} |\Psi_{k,i,p}\rangle} \wedge |z_{k,p-1,j}\rangle\right) \oplus |z_{k,p,j}\rangle$
      end if
    end for
  end for
end for

Fig. 4. Algorithm for summing up energy.

each conformation, the algorithm extracts the values from vector $|\Psi\rangle$ and stores their sum in the state

$$|z\rangle = \bigotimes_{j=4(N-4)+3}^{0} \left|z_{N-3,3,j}\right\rangle \tag{17}$$

The algorithm starts with the first amino acid ($k = 1$) in a sequence. It checks whether the amino acid has any neighbors in the lattice ($\left|\Psi_{1,i,p}\right\rangle = |1\rangle$ for at least one i). Starting from the northern neighbor ($p = 0$), if such a neighbor exists, $\left|z_{1,0,1}\right\rangle$ will be equal to $|1\rangle$. On the other hand, if for example the first amino acid has a western neighbor ($p = 3$), $\left|z_{1,3,1}\right\rangle$ will be equal to $|1\rangle$. If the first amino acid has both neighbors however, this fact will be indicated by the state $\left|z_{1,3,2}\right\rangle = |1\rangle$, with the last index $j = 2$ standing for the number of neighbors. The presence of any other neighbors for the first and the consecutive amino acids will be indicated by the single digit 1 located at the position j in the final string $\left|z_{N-3,3,j}\right\rangle$.

Every such string $|z\rangle$ contains exactly one digit 1 with all the remaining digits being 0. To illustrate the principle on an example: a conformation having four hydrophobic lattice contacts ($j = 4$) will have the digit 1 placed on the 5th position counting from the right ($|0\cdots010000\rangle$). The closer to the left the digit 1 is located, the higher the number of hydrophobic lattice contacts. In accordance with the free energy scoring function ($V = -|(a_i, a_j)|$), conformations with the digit 1 located closest to the left have the lowest free energy.

The algorithm in Fig. 4 makes use of quantum gates for logical conjunction (AND) and disjunction (OR), which are based on the Toffoli gate [55]. For the implementation of logical conjunction and disjunction operations see Fig. 5(a). Each conjunction and each disjunction require one auxiliary qubit, either in the $|0\rangle$ or the $|1\rangle$ state. As shown in Fig. 5(b), a total of $N-2$ such auxiliary qubits $|r\rangle$ are needed to compute each disjunctive clause. Then a further 2 qubits, as indicated in Fig. 5(c), must be used to carry out the remaining operations in the algorithm in Fig. 4. Both the $|r\rangle$ qubits as well as the other 2 auxiliary qubits can be reset to their original state after the state $\left|z_{k,p,j}\right\rangle$ has been calculated.

4.5. Identifying conformations with minimal energy

Having found the energy values for each conformation, the conformation(s) for which the free energy is the lowest must be found. This can be done e.g. with Grover's search algorithm [28]. To that end, the amplitudes of vector $|z\rangle$ in (17) must subsequently be amplified in order to find a conformation with the highest number of hydrophobic lattice contacts. Grover's algorithm performs optimally when run $\frac{\pi}{4}\sqrt{n/m}$ iterations, where m is the number of solutions and $n = 2^{2(N-1)}$ is the number of conformations. If necessary, the number of solutions m can be determined with the quantum counting algorithm [15]. Both the search algorithm and the counting algorithm require $O(\sqrt{n/m})$ calls to Grover iteration and thus $O(\sqrt{n/m})$ oracle calls. Upon measurement, the circuit outputs a single conformation that is a solution for the particular amino acid sequence. Multiple runs of the algorithm are needed in order to output all solutions.

The oracle for our algorithm involves finding the maximal element among states $\left|z_{N-3,3,j}\right\rangle$. In order to do that, we propose a checking function in the form of a string $|s\rangle$ of the same length as the elements $\left|z_{N-3,3,j}\right\rangle$, i.e., $4(N-3)$. The oracle would compare $|s\rangle$ with all $\left|z_{N-3,3,j}\right\rangle$ entries by means of bit-wise exclusive disjunction operation to determine whether a given $\left|z_{N-3,3,j}\right\rangle$ is a maximal element:

$$\bigotimes_j \left|s_j\right\rangle = \bigotimes_j \left|z_{N-3,3,j} \oplus s_j\right\rangle \tag{18}$$

where $0 \le j \le 4(N-4)+3$. This operation requires only linear time as the comparisons are carried out in superposition. Subsequently, by applying logical disjunction on vector $|s\rangle$ and then negating the result

$$|s\rangle = \neg\left(\bigvee\left(\bigotimes_j \left|s_j\right\rangle\right)\right) \tag{19}$$

the output is $|s\rangle = |1\rangle$ if and only if the compared strings match. A vector in $|z\rangle$ is maximal if and only if the operation in (18) results in $\bigotimes_j |s_j\rangle = |0\rangle^{\otimes 4(N-3)}$. Since the oracle has to be in a superposition state, $|s\rangle$ must be integrated into the state vector prior to computation (at the time of initializing the system). String $|s\rangle$ might need to be decremented and the computation runs again till a conformation with minimal free energy is found. In the worst case, decrementing a total of $4(N-3)$ times is needed, which contributes a linear factor to the complexity.

5. Complexity assessment

The assessment of both the temporal and the spatial complexity is straightforward. Given an amino acid sequence of length N, the required number of qubits, including those considered as auxiliary, is $16N^3 - 24N^2 - 12N + 63$, implying a spatial asymptotic complexity $O(N^3)$, which is a polynomial increment with N. As stated previously in our Introduction, the conformational space cannot be created classically due to lack of classical computational resources.

With regards to the temporal complexity, the construction of conformational space takes $(N-1)$ iterations with $2\theta - 4$ NOT, $7\theta - 3$ CNOT, and $2\theta - 1$ Toffoli gates in each iteration, but the number of time steps is only $(N-1)(2\theta+4)$. Finding free energies takes $4(N-3)^2$ iterations. Each iteration implements $5\theta + 2$ NOT, $11\theta + 1$ CNOT, and $5\theta - 3$ Toffoli gates. Summing up energies takes $8(N-3)^2 + 2(N-3)$ iterations with $4(N-3)+1$ NOT, $N-1$ CNOT and 2 Toffoli gates. The above operations together have the asymptotic complexity of $O(N^3)$.

In the case that Grover's algorithm is used to identify solutions, which are the conformations with the minimal free energy, the complexity is $O(N^3\sqrt{2^{2(N-1)}})$ which reduces to the asymptotic complexity $O(N^3\sqrt{2^N})$.

Given the above, protein structure prediction in two-dimensional hydrophobic-hydrophilic model on a square lattice can be solved by our quantum algorithm with a quadratic speedup in comparison with what a classical algorithm can currently achieve.

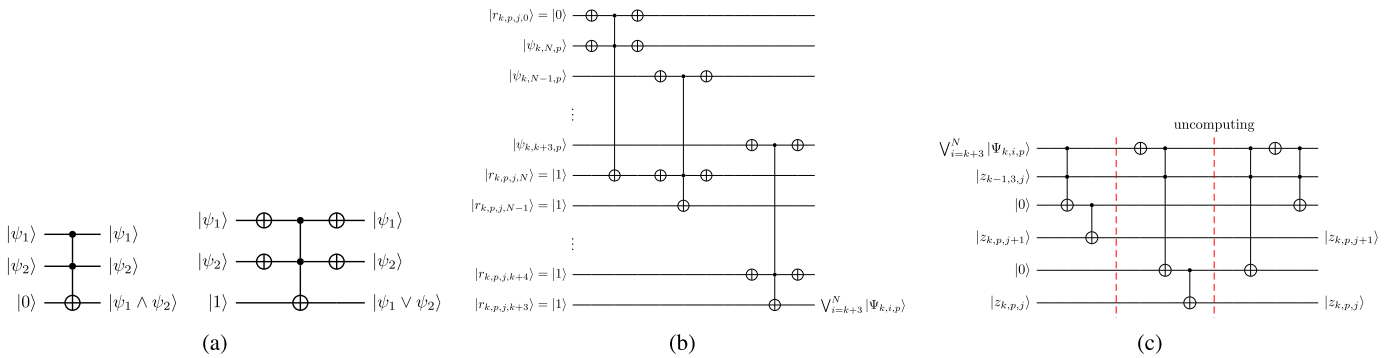

Fig. 5. (a) Quantum circuit for a disjunctive (left) and conjunctive (right) clauses. (b) Quantum circuit for the OR clause in rows 5 and 8 in Fig. 4. This is essentially also the implementation of the OR clauses in rows 6 and 9, with the difference that these two clauses require a negation as well, which can be implemented with a Pauli-X gate. (c) Quantum circuit for the remaining operations in Fig. 4.

6. Experimental validation of the algorithm

An exact implementation of the quantum algorithm for protein structure prediction in two-dimensional hydrophobic-hydrophilic model requires around 200 logical qubits, an amount that is currently not available on any kind of quantum devices. For that reason, we validate a simplified instance of the algorithm and test it on IBM qasm simulator which allows for testing of circuits of up to 32 qubits. Our instance aims to test the performance of the algorithm on amino acid sequences of length $L = 3$ and involves 25 qubits (instances for amino acid sequences of length $L \geq 4$ would exceed the limit of 32 qubits). The number of bits to represent the coordinates in two's complement notation is therefore $t = \lceil \log_2(2N-1) \rceil = 3$. The qubits are allocated as follows (here we also include information on states omitted from the exact description of our algorithm for PSP):

- The sequence is not encoded, instead it is implicitly assumed to be $|101\rangle$, where $|1\rangle$ stands for a hydrophobic and $|0\rangle$ stands for a hydrophilic amino acid. The first and the last amino acids in the sequence are hence hydrophobic while the amino acid in the middle is hydrophilic.
- As the length is $N = 3$, two qubits encode the directional transition from the first to the second amino acid, and a further two qubits encode the transition from the second to the third amino acid. These four qubits correspond to vector $|w\rangle$.
- The coordinates $|x_1\rangle$ and $|y_1\rangle$ of the first amino acid are assumed to be $|000\rangle$. They are not encoded in the circuit.
- Three qubits are needed to encode the four coordinates: the $|x_2\rangle$ coordinate of amino acid 2, the $|y_2\rangle$ coordinate of amino acid 2, the $|x_3\rangle$ coordinate of amino acid 3, and the $|y_3\rangle$ coordinate of amino acid 3. This gives a total of 12 qubits.
- Three qubits encode the decimal 1 $|\alpha\rangle = |001\rangle$. The two's complement of 1 (-1) is not encoded. It will be generated from $|\alpha\rangle$ when necessary by negating the first two qubits.
- One qubit encodes the carry qubit $|c\rangle$ for our implementation of doubly-controlled quantum adder.
- One ancillary qubit $|g\rangle$ encodes the intermediate value needed in the Grover's diffusion operator.
- One qubit $|e\rangle$ encodes the energy value for each conformation.
- Three ancillary qubits $|anc\rangle$ are used whenever necessary in the quantum adder circuit, in Grover's oracle, and in the diffusion operator.

With this, the quantum state of the simplified instance is:

$$|w\rangle\,|x\rangle\,|y\rangle\,|\alpha\rangle\,|c\rangle\,|g\rangle\,|e\rangle\,|anc\rangle \tag{20}$$

This state is then subdivided as follows:

$$|w_1 w_2\rangle\,|x_2 x_3\rangle\,|y_2 y_3\rangle\,|\alpha\rangle\,|c\rangle\,|g\rangle\,|e\rangle\,|anc\rangle \tag{21}$$

where $|w_1\rangle$ encodes the transition direction from the first amino acid to the second amino acid in a sequence, while $|w_2\rangle$ encodes the transition direction from the second to the third amino acid in the sequence. $|x_2\rangle$ stands for the x coordinate of the second amino acid, $|x_3\rangle$ stands for the x coordinate of the third amino acid, $|y_2\rangle$ stands for the y coordinate of the second amino acid, and $|y_3\rangle$ stands for the y coordinate of the third amino acid. The coordinates are encoded in two's complement notation to facilitate straightforward addition and subtraction.

There are four transition directions for a square lattice as given below, where $i = 1, 2$:

- $|w_i\rangle = |00\rangle$ stands for the transition northward and it entails an increase by 1 of the y coordinate of the $(i+1)$st amino acid: $|y_{i+1}\rangle = |y_i\rangle + |001\rangle$
- $|w_i\rangle = |01\rangle$ stands for the transition eastward and it entails an increase by 1 of the x coordinate of the $(i+1)$st amino acid: $|x_{i+1}\rangle = |x_i\rangle + |001\rangle$
- $|w_i\rangle = |10\rangle$ stands for the transition southward and it entails a decrease by 1 of the y coordinate of the $(i+1)$st amino acid: $|y_{i+1}\rangle = |y_i\rangle + |111\rangle$. Here $|111\rangle = -1$ is the two's complement of 1.
- $|w_i\rangle = |11\rangle$ stands for the transition westward and it entails a decrease by 1 of the x coordinate of the $(i+1)$st amino acid: $|x_{i+1}\rangle = |x_i\rangle + |111\rangle$. Here $|111\rangle = -1$ is the two's complement of 1.

The square lattice requires a sequence of at least four amino acids in order to make an adjacent hydrophobic-hydrophobic contact in the lattice but not in the sequence. As the available quantum resources do not allow for an implementation of our algorithm on sequences of length four and above (encoding for such sequences exceeds the limit of 32 qubits), we conduct two simulations on a simplified form of our algorithm.

In each of the two simulations we assume that the length of the sequence is $L = 3$. The difference between them is only in the selected conformation (solution): $|w\rangle = |1010\rangle$ or $|1100\rangle$. The solutions are selected arbitrarily and can be changed arbitrarily without affecting the performance of our algorithm.

In the following we will assume that the algorithm makes use of Grover's quantum search algorithm. Grover's algorithm finds a solution to a given problem with a high probability and is optimal for the problem. The structure of the PSP algorithm is given in Fig. 6.

Each of the simplified instances of the algorithm consists of the following steps:

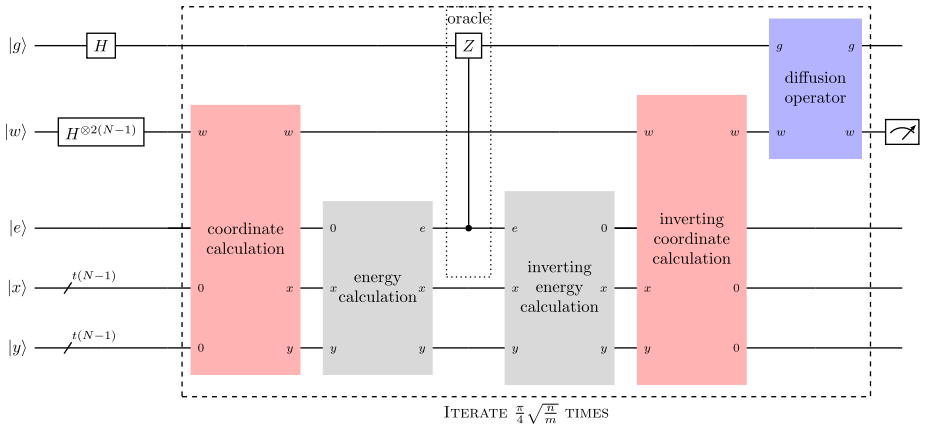

Fig. 6. Circuit schematic for PSP algorithm using Grover's amplitude amplification routine. N is the length of the amino acid sequence, while t stands for the length of a single coordinate in bits. Inside the operation boxes we indicate which parameters (out of w, x, y, e and g) participate in the given operation. A value 0 indicates that the respective state is reset to 0. A letter indicates that the respective parameter is in its calculated form.

1. Register initialization, including setting the system into a superposition over vectors $|w\rangle$ and $|g\rangle$.
2. Calculation of the two-dimensional Cartesian coordinates $|x\rangle$ and $|y\rangle$ for each conformation.
3. Calculation of the energy values for each conformation. The energy value is stored in quantum register $|e\rangle$. $|e\rangle = |1\rangle$ if and only if the respective conformation is a solution. Otherwise, $|e\rangle = |0\rangle$.
4. Inverting the phase of the solution. This is done by flipping the phase of each state $|g\rangle = |1\rangle$ whenever $|e\rangle = |1\rangle$. For $|e\rangle = |0\rangle$, no phase inversion takes place.
5. Resetting the energies of each conformation. This step is required for the proper functioning of Grover's diffusion operator in step 7.
6. Resetting the coordinates of all conformations by executing the coordinate calculation step in reverse. This step results in all coordinates being reset to all-zero values and it is required for the correct performance of Grover's algorithm in step 7.
7. Application of Grover's diffusion operator to registers $|w\rangle$ and $|g\rangle$.

Steps 2 through 7 are iterated over as many times as the Grover algorithm requires for a given number of solutions and candidate conformations. The required number of iterations is calculated from the formula

$$\frac{\pi}{4}\sqrt{\frac{n}{m}} \tag{22}$$

where $n = 32$ is $2\times$ the number of all candidate conformations and $m = 1$ is the number of solutions. The number of candidate conformations for length $L = 3$ is $\frac{n}{2} = 16$. Factor 2 comes from the fact that vector $|g\rangle$ is also in superposition.

The schematic in Fig. 6 shows these seven steps as elements of a quantum circuit.

6.1. Setting the state into a superposition

Step 1 is straightforward. The quantum system is set into a uniform superposition by applying Hadamard gates to the four qubits in register $|w\rangle$ and the single qubit in register $|g\rangle$. Omitting all other quantum registers for the sake of clarity, the resulting state is

$$\frac{1}{\sqrt{2^{2(N-1)}}}\sum_{w=0}^{2^{2(N-1)}-1}|w\rangle \otimes \frac{1}{\sqrt{2}}\sum_{g=0}^{1}|g\rangle \tag{23}$$

indicating that for sequences of length $L = 3$, the total number of conformations is $2^{2(L-1)} = 16$, as $|g\rangle$ is not part of a conformation. The candidate conformations for a native conformation are $|w\rangle = |0000\rangle$, $|0001\rangle$, $|0010\rangle$, …, through $|1111\rangle$. Each of these states is to be understood as composed of two values. The first two bits represent the directional transition from the first amino acid to the second. The last two bits represent the directional transition from the second amino acid to the third. For instance, $|0001\rangle$ consists of $|00\rangle$ corresponding to the transition northwards from the first to the second amino acid, and of $|01\rangle$ corresponding to the transition to the eastwards from the second to the third amino acid.

6.2. Coordinate calculation

In step 2 the respective coordinates for each conformation are calculated depending on the value of vector $|w\rangle$. The coordinates are stored in two's complement notation which facilitates straightforward addition and subtraction. Our calculation of coordinates relies on a doubly controlled quantum adder that we adapted from [20] by incorporating two control qubits into it.

6.3. Energy calculation and oracle

In step 3 of this simplified experimental validation the energy value for each conformation is calculated simultaneously in superposition. We arbitrarily choose two different conformations to be treated as solutions: $|1010\rangle$ and $|1100\rangle$. The conformations are chosen arbitrarily because for $N = 3$ no actual solution exists. The structure of the square lattice dictates that a solution exists only for sequences of length $N \geq 4$. For $N = 4$ some conformations will have a single loose contact, for $N < 4$ no loose contact will ever be present.

Below we describe the energy calculation operation and the oracles for the two solutions.

A. Identifying conformation $|w\rangle = |1010\rangle$ The conformation $|1010\rangle$ is shown in Fig. 7(a). As $|w_1\rangle = |10\rangle$ and $|w_2\rangle = |10\rangle$, this conformation is generated by taking two steps southwards, which affects only the y coordinates of the second and the third amino acids. The energy calculation and the oracle that selects this conformation is given in Fig. 7(b). The coordinates for this conformation are $|x_1\rangle|x_2\rangle|x_3\rangle = |000\rangle|000\rangle|000\rangle$ and $|y_1\rangle|y_2\rangle|y_3\rangle = |000\rangle|111\rangle|110\rangle$. We note that $|111\rangle = -1$ and $|110\rangle = -2$ in two's complement notation. As it was not possible to implement the full code of the algorithm, certain simplifications had to be made, which however do not affect the general functionality of

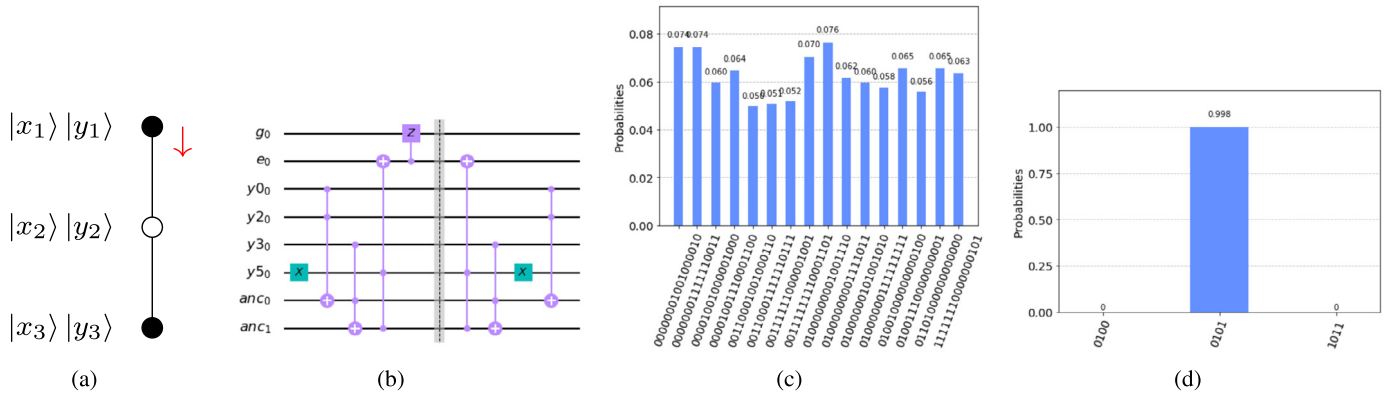

Fig. 7. Experimental outputs for the conformation $|w\rangle = |1010\rangle$. Histograms are to be read from top to bottom. (a) Conformation $|w\rangle = |1010\rangle$. (b) The oracle for selecting the conformation $|1010\rangle$. The gates before the barrier are for calculating the energy and inverting the phase, while the gates behind the barrier are for uncomputing the energy. (c) The result of calculating coordinates and energy. Bits 1 through 4 represent the conformation $|w\rangle$. Bits 5 through 7 represent the $|x_2\rangle$ coordinate, bits 8 through 10 the $|x_3\rangle$ coordinate, bits 11 through 13 the $|y_2\rangle$ coordinate, and bits 14 though 16 the $|y_3\rangle$ coordinate. The last bit stands for the energy $|e\rangle$. (d) The probability of observing the conformation $|w\rangle = |1010\rangle$ upon measurement is 0.9998.

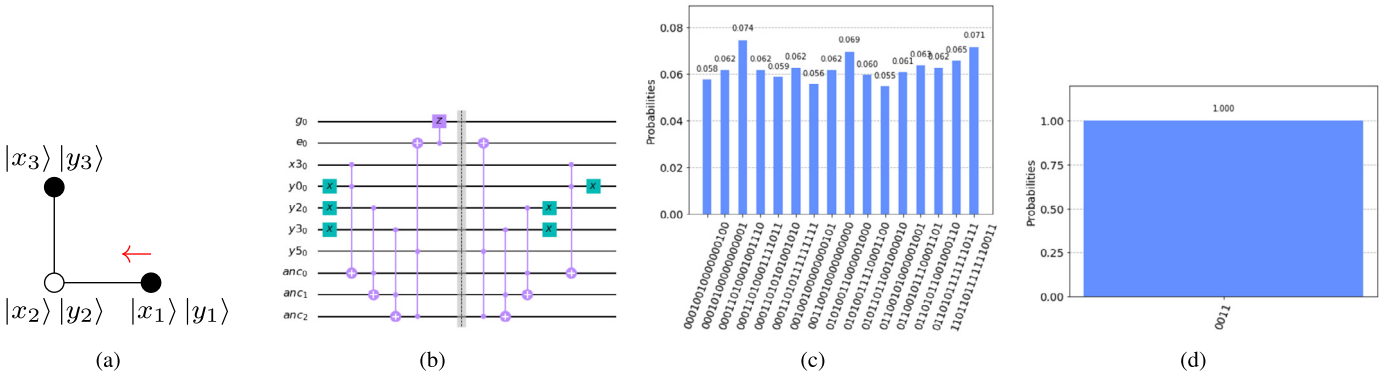

Fig. 8. Experimental outputs for the conformation $|w\rangle = |1100\rangle$. Histograms are to be read from top to bottom. (a) Conformation $|w\rangle = |1100\rangle$. (b) The oracle for selecting the conformation $|1100\rangle$. The gates before the barrier are for calculating the energy and inverting the phase, while the gates behind the barrier are for uncomputing the energy. (c) The result of calculating coordinates and energy. Bits 1 through 4 represent the conformation $|w\rangle$. Bits 5 through 7 represent the $|x_2\rangle$ coordinate, bits 8 through 10 the $|x_3\rangle$ coordinate, bits 11 through 13 the $|y_2\rangle$ coordinate, and bits 14 though 16 the $|y_3\rangle$ coordinate. The last bit stands for the energy $|e\rangle$. (d) The probability of observing the conformation $|w\rangle = |1100\rangle$ upon measurement is 1.000.

the algorithm. Upon analysis, we observe that this conformation is uniquely identified by the $|y_2\rangle\,|y_3\rangle$ component. Namely, by checking that the first and the third bit in both $|y_2\rangle$ and $|y_3\rangle$ are, respectively, $|y0\rangle = |1\rangle, |y2\rangle = |1\rangle, |y3\rangle = |1\rangle, |y5\rangle = |0\rangle$, we can determine whether this is the conformation we are looking for. In Fig. 7(b), the energy is stored in qubit $|e\rangle$. If, upon coordinate checking, it is found that it is the desired conformation, $|e\rangle$ changes its value to $|1\rangle$. Otherwise, it stays $|0\rangle$. Fig. 7(c) shows correctly calculated coordinates and energies.

B. Identifying conformation $|w\rangle = |1100\rangle$ The conformation $|1100\rangle$ is shown in Fig. 8(a). As $|w_1\rangle = |11\rangle$ and $|w_2\rangle = |00\rangle$, this conformation is generated by taking first one step westwards and then another step northwards. The energy calculation and the oracle that selects this conformation is given in Fig. 8(b). The coordinates for this conformation are $|x_1\rangle\,|x_2\rangle\,|x_3\rangle = |000\rangle\,|111\rangle\,|111\rangle$ and $|y_1\rangle\,|y_2\rangle\,|y_3\rangle = |000\rangle\,|000\rangle\,|001\rangle$. As it was not possible to implement the full code of the algorithm, certain simplifications had to be made, which however do not affect the general functionality of the algorithm. Upon analysis, we observe that this conformation is uniquely identified by the $|x_3\rangle$ and the $|y_2\rangle\,|y_3\rangle$ components. Hence, we only need to check the first bit of $|x_3\rangle$, as well as the first and the third bit in both $|y_2\rangle$ and $|y_3\rangle$. These five values are, respectively, $|x3\rangle = |1\rangle$, $|y0\rangle = |0\rangle$, $|y2\rangle = |0\rangle$, $|y3\rangle = |0\rangle$, $|y5\rangle = |1\rangle$. In Fig. 8(b), the energy is stored in qubit $|e\rangle$. If, upon coordinate check, it is found that it is the desired conformation, $|e\rangle$ changes its value to $|1\rangle$. Otherwise, it stays $|0\rangle$. Fig. 8(c) shows correctly calculated coordinates and energies.

Fig. 7(b) and Fig. 8(b) also show how the energy qubit $|e\rangle$ is reset by executing the energy calculation in reverse order. These reverse operations are indicated behind the barrier lines.

6.4. Resetting coordinates

Next, in step 6, the coordinates calculated in step 2 are uncomputed by executing the coordinate calculation protocol in reverse order. After this step, all coordinates of all conformations are reset to all-zero values. At this point, only $|w\rangle$ and $|g\rangle$ contain non-constant values, while all other qubits are either set to $|0\rangle$ or $|1\rangle$ for each conformation. The quantum state for the conformations for which $|e\rangle = |1\rangle$ is

$$\frac{1}{\sqrt{2^{2(N-1)}}} \sum_{w=0 \wedge e=1}^{2^{2(N-1)}-1} |w\rangle\,|-\rangle_g\,|0\rangle_e\,|0\rangle_x^{\otimes 9}\,|0\rangle_y^{\otimes 9} \tag{24}$$

while for all other conformations, i.e. those for which $|e\rangle = |0\rangle$ is

$$\frac{1}{\sqrt{2^{2(N-1)}}} \sum_{w=0 \wedge e=1}^{2^{2(N-1)}-1} |w\rangle\,|+\rangle_g\,|0\rangle_e\,|0\rangle_x^{\otimes 9}\,|0\rangle_y^{\otimes 9} \tag{25}$$

where $|-\rangle = \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$, $|+\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)$.

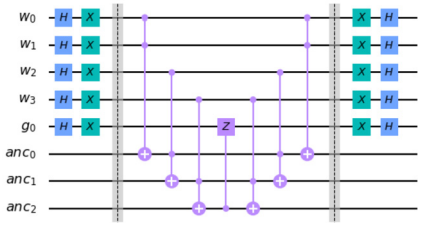

Fig. 9. The diffusion operator (the same for both simulations).

Table 2
Required quantum gates.

Gates (per iteration)	$\|1010\rangle$	$\|1100\rangle$
Hadamard gates	15	15
Pauli-X gates	61	65
Controlled-X (CX) gates	24	24
Toffoli (CCX) gates	716	718
Controlled-Z (CZ) gates	2	2
TOTAL (4 iterations)	3,272	3,296

6.5. Applying diffusion operator

In the last step, i.e., step 7, the diffusion operator is applied to $|w\rangle$ and $|g\rangle$. This operator is the same for any amino acid sequence of length $N = 3$ and is shown in Fig. 9. It implements an inversion about the quantum state. Our implementation of the operator makes use of three ancillary qubits to store intermediate results.

6.6. Measurement

The selected conformations $|1010\rangle$ or $|1100\rangle$ will be output upon measuring the $|w\rangle$ quantum register. As shown in Fig. 7(d), the probability of observing the conformation $|1010\rangle$ is 0.998. Fig. 8(d) shows the probability of observing the conformation $|1100\rangle$ to be 1.

7. Required quantum resources

The two experiments were coded in Python and carried out on IBM qasm simulator. The required number of basic quantum gates is as given in Table 2 for each of the two conformations tested.

8. Code availability

The Jupyter Notebook with the Python source code will be provided by the corresponding author upon request. The code uses IBM Quantum's library QISKit [5] to render quantum circuits. The code can be executed directly in IBM Quantum's Lab [32] or upon a local installation of Qiskit.

9. Conclusion

While it is impossible to solve the PSP problem in the HP model classically, it will be possible once quantum computers become fault-tolerant and large enough to allow for execution of large quantum circuits. As we show in this paper, a quantum counterpart of the classical algorithm performs better with respect to both time and space. In order to store all possible conformations, calculate their coordinates and energy, and finally select the minimum energy conformation, only a number of qubits that is polynomial ($O(N^3)$) in the input size is required, where N is the number of amino acids in the input sequence. With respect to time, a quadratic speedup can be achieved compared to the classical algorithm.

In this paper, we use quantum computational principles of superposition, entanglement and destructive/constructive amplitude interference to solve the problem of predicting the protein structure in the two-dimensional hydrophobic-hydrophilic model on a square lattice. The algorithm that we propose is, to our best knowledge, the first quantum algorithm for PSP that is fully specified down to quantum gates. The algorithm correctly predicts the solution, as defined under the HP model, and has a high probability of outputting the solution upon a measurement (see Fig. 7(b) and Fig. 7(d)). This high probability is corroborated by its theoretical estimation (see Appendix A).

Quantum computing is an emerging field. The presently available state-of-the-art quantum resources are limited. Most of the freely available quantum devices provide up to 14 qubits, some lab developments include up to 79 qubits [31,47,27,7]. Furthermore, the devices are not yet fault-tolerant to the extent of facilitating reliable, purposeful computation at low error rates. In fact, with more qubits added and with increasing circuit depth, the dynamics of quantum devices becomes intractable [18]. Therefore, as an intermediate solution, IBM's quantum simulator provides a platform for reliable testing of quantum circuits of up to 32 qubits. The experimental evaluation of our algorithm was carried out on this simulator. Given the constraints on the qubit number, three amino acids were the maximal size that could be simulated and required 25 qubits. Our circuit depths reached 3,272 and 3,296 quantum gates, respectively, for the two simulations. As such, these are some of the largest quantum circuits ever simulated.

Funding sources

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

CRediT authorship contribution statement

Renata Wong: Conceptualization, Methodology, Software, Validation, Writing – original draft. **Weng-Long Chang:** Conceptualization, Methodology, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Theoretical probability of finding the minimum

The five qubits that are the inputs to Grover's routine are $|w\rangle$ and $|g\rangle$. Let $|k\rangle$ stand for $|w\rangle|g\rangle$. The initial state is

$$|\psi_0\rangle = \frac{1}{\sqrt{32}}|k^*\rangle + \sum_{|k\rangle \neq |k^*\rangle} \frac{1}{\sqrt{32}}|k\rangle \tag{A.1}$$

where $|k^*\rangle$ is the single solution, i.e. the conformation that should be output after executing the PSP algorithm. The first application of the oracle negates the phase of the solution while leaving the phases of all other states intact:

$$|\psi_1\rangle = -\frac{1}{\sqrt{32}}|k^*\rangle + \sum_{|k\rangle \neq |k^*\rangle} \frac{1}{\sqrt{32}}|k\rangle \tag{A.2}$$

The mean is given as

$$\mu = \frac{1}{32}\sum_k \alpha_k \tag{A.3}$$

Its value in the first iteration of the algorithm is $\mu_1 = 0.1657$. The diffusion operator calculates

$$\sum_k (2\mu - \alpha_k) |k\rangle \tag{A.4}$$

and results in the following state for the first iteration:

$$|\psi_3\rangle = \frac{92}{181} |k^*\rangle + \sum_{|k\rangle \neq |k^*\rangle} \frac{28}{181} |k\rangle \tag{A.5}$$

This corresponds to the probability of outputting the solution upon measurement being 0.2583.

Following this procedure, at the end of the second iteration the probability of detecting the solution upon measurement is 0.6025. At the end of the third iteration, the probability of finding the solution is 0.8971. At the end of the fourth iteration of the algorithm, the probability of seeing the solution is 0.9993. This value corresponds closely to the one obtained by us in Fig. 7(d) and Fig. 8(d).

References

[1] S. Aaronson, https://scottaaronson.blog/?p=1400, 2013, May 16.
[2] S. Aaronson, A. Arkhipov, The computational complexity of linear optics, in: Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing, ACM, San Jose, 2011, pp. 333–342.
[3] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, Adiabatic quantum computation is equivalent to standard quantum computation, SIAM J. Comput. 37 (2007) 166.
[4] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, Peter Walter, Molecular Biology of the Cell, Garland Science, New York, 2002.
[5] G. Aleksandrowicz, et al., Qiskit: an open-source framework for quantum computing, Tech. Rep., 2019.
[6] M. Amico, Z.H. Saleem, M. Kumph, An experimental study of Shor's factoring algorithm on IBM Q, Phys. Rev. A 100 (1) (2019) 012305.
[7] F. Arute, et al., Quantum supremacy using a programmable superconducting processor, vol. 574, Nature (2019) 505–510.
[8] R. Babbush, A. Perdomo-Ortiz, B. O'Gorman, W. Macready, A. Aspuru-Guzik, Construction of energy functions for lattice heteropolymer models: efficient encodings for constraint satisfaction programming and quantum annealing, Adv. Chem. Phys. 155 (2014) 201–244.
[9] T. Babej, C. Ing, M. Fingerhuth, Coarse-grained lattice protein folding on a quantum annealer, arXiv:1811.00713, 2018.
[10] P.K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, S. Woerner, Improving variational quantum optimization using CVaR, Quantum 4 (2020) 256.
[11] C. Bennett, E. Bernstein, G. Brassard, U. Vazirani, Strengths and weaknesses of quantum computing, SIAM J. Comput. 26 (5) (1997) 1510–1523.
[12] S. Boixo, S.V. Isakov, V.N. Smelyanskiy, et al., Characterizing quantum supremacy in near-term devices, Sci. Phys. 14 (2018) 595–600.
[13] S. Boixo, S.V. Isakov, V.N. Smelyanskiy, H. Neven, Simulation of low-depth quantum circuits as complex undirected graphical models, arXiv:1712.05384v2, 2018.
[14] M. Born, V.A. Fock, Beweis des Adiabatensatzes, Z. Phys. A 51 (3–4) (1928) 165–180.
[15] G. Brassard, P. Hoyer, A. Tapp, Quantum counting, in: Automata, Languages and Programming, Springer, Berlin Heidelberg, 1998, pp. 820–831.
[16] W.L. Chang, Q. Yu, Z.K. Li, J.H. Chen, X.H. Peng, M. Feng, Quantum speedup in solving the maximal-clique problem, Phys. Rev. A 97 (2018) 032344.
[17] Zhao-Yun Chen, Qi Zhou, Cheng Xue, Xia Yang, Guang-Can Guo, Guo-Ping Guo, 64-qubit quantum circuit simulation, Sci. Bul. 63 (15) (2018) 964–971.
[18] A.D. Corcoles, A. Kandala, A. Javadi-Abhari, D.T. McClure, A.W. Cross, K. Temme, P.D. Nation, M. Steffen, J.M. Gambetta, Challenges and opportunities of near-term quantum computing systems, Proc. IEEE 108 (8) (2019) 1338–1352.
[19] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, M. Yannakakis, On the complexity of protein folding, J. Comput. Biol. 5 (3) (1998) 423–465.
[20] S.A. Cuccaro, T.G. Draper, S.A. Kutin, D. Petrie Moulton, A new quantum ripple-carry addition circuit, arXiv:quant-ph/0410184, 2004.
[21] K.A. Dill, Theory for the folding and stability of globular proteins, Biochemistry 24 (6) (1985) 1501–1509.
[22] S.P. Dubey, N.G. Kini, S. Balaji, M.S. Kumar, A review of protein structure prediction using lattice model, Crit. Rev. Biomed. Eng. 46 (2) (2018) 147–162.
[23] E. Farhi, J. Goldstone, S. Gutman, M. Sipser, Quantum computation by adiabatic evolution, arXiv:quant-ph/0001106v1, 2000.
[24] E. Farhi, J. Goldstone, S. Gutmann, A quantum approximate optimization algorithm, arXiv:1411.4028v1, 2014.
[25] C. Figgatt, D. Maslov, K.A. Landsman, N.M. Linke, S. Debnath, C. Monroe, Complete 3-qubit Grover search on a programmable quantum computer, Nat. Commun. 8 (1) (2017), 1918.
[26] M. Fingerhuth, T. Babej, C. Ing, A quantum alternating operator ansatz with hard and soft constraints for lattice protein folding, arXiv:1810.13411, 2018.
[27] Google, https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html.
[28] L.K. Grover, Quantum computers can search rapidly by using almost any transformation, Phys. Rev. Lett. 80 (19) (1998) 4329–4332.
[29] L.K. Grover, Synthesis of quantum superpositions by quantum computation, Phys. Rev. Lett. 85 (6) (2000) 1334–1337.
[30] A.W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. 103 (15) (2009) 150502.
[31] IBM, https://www.research.ibm.com/ibm-q/technology/systems/.
[32] IBM Quantum Lab, https://quantum-computing.ibm.com/.
[33] M. Jiang, B. Zhu, Protein folding on the hexagonal lattice in the HP model, J. Bioinform. Comput. Biol. 3 (1) (2005) 19–34.
[34] S. Jiang, K.A. Britt, A.J. McCaskey, T.S. Humble, S. Kais, Quantum annealing for prime factorization, Sci. Rep. 8 (1) (2018) 17667.
[35] A. Kandala, K.X. Wei, S. Srinivasan, E. Magesan, S. Carnevale, G.A. Keefe, D. Klaus, O. Dial, D.C. McKay, Demonstration of a high-fidelity cnot gate for fixed-frequency transmons with engineered ZZ suppression, Phys. Rev. Lett. 127 (2021) 130501.
[36] B.P. Lanyon, T.J. Weinhold, N.K. Langford, M. Barbieri, D.F.V. James, A. Gilchrist, A.G. White, Experimental demonstration of a compiled version of Shor's algorithm with quantum entanglement, Phys. Rev. Lett. 99 (25) (2007) 250505.
[37] J.C. Loredo, M.A. Broome, P. Hilaire, O. Gazzano, I. Sagnes, A. Lemaitre, M.P. Almeida, P. Senellart, A.G. White, Boson sampling with single-photon Fock states from a bright solid-state source, Phys. Rev. Lett. 118 (13) (2017) 130503.
[38] E. Martín-López, A. Laing, T. Lawson, R. Alvarez, X.Q. Zhou, J.L. O'Brien, Experimental realization of Shor's quantum factoring algorithm using qubit recycling, Nat. Photonics 6 (11) (2012) 773–776.
[39] S. Miyazawa, 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 (1996) 623–644.
[40] T. Monz, D. Nigg, E.A. Martinez, M.F. Brandl, P. Schindler, R. Rines, S.X. Wang, I.L. Chuang, R. Blatt, Realization of a scalable Shor algorithm, Science 351 (6277) (2016) 1068–1070.
[41] A. Onofrio, G. Parisi, G. Punzi, S. Todisco, M.A. Di Noia, F. Bossis, A. Turi, A. De Grassi, C.L. Pierri, Distance-dependent hydrophobic-hydrophobic contacts in protein folding simulations, Phys. Chem. Chem. Phys. 16 (35) (2014) 18907–18917.
[42] E. Pednault, J.A. Gunnels, et al., Pareto-efficient quantum circuit simulation using tensor contraction deferral, arXiv:1710.05867v4, 2020.
[43] X.H. Peng, Z.Y. Liao, N.Y. Xu, G. Qin, X.Y. Zhou, D. Suter, J.F. Du, Quantum adiabatic algorithm for factorization and its experimental implementation, Phys. Rev. Lett. 101 (22) (2008) 220405.
[44] A. Perdomo, C. Truncik, I. Tubert-Brohman, G. Rose, A. Aspuru-Guzik, Construction of model Hamiltonians for adiabatic quantum computation and its application to finding low-energy conformations of lattice protein models, Phys. Rev. A 78 (1) (2008) 012320.
[45] A. Perdomo-Ortiz, N. Dickson, M. Drew-Brook, G. Rose, A. Aspuru-Guzik, Finding low-energy conformations of lattice protein models by quantum annealing, Sci. Rep. 2 (2012) 571.
[46] A. Peruzzo, J. McClean, et al., A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. 5 (2014) 4213.
[47] Rigetti, https://www.rigetti.com/.
[48] A. Robert, P. Barkoutsos, S. Woerner, I. Tavernelli, Resource-efficient quantum algorithm for protein folding, npj Quantum Inf. 7 (2021) 38.
[49] Y.R. Sanders, G.H. Low, A. Scherer, D.W. Berry Black, Box quantum state preparation without arithmetic, Phys. Rev. Lett. 122 (2) (2019) 020502.
[50] M.O. Scully, M.S. Zubairy, Quantum optical implementation of Grover's algorithm, Proc. Natl. Acad. Sci. 98 (17) (2001) 9490–9493.
[51] P. Shor, Progress in quantum algorithms, in: H.O. Everitt (Ed.), Experimental Aspects of Quantum Computing, Springer US, 2005, pp. 5–13.
[52] P.W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: Proceedings 35th Annual Symposium on Foundations of Computer Science, IEEE Comput. Soc. Press, 1994, pp. 124–134.
[53] J.B. Spring, B.J. Metcalf, P.C. Humphreys, W.S. Kolthammer, X.M. Jin, M. Barbieri, A. Datta, N. Thomas-Peter, N.K. Langford, D. Kundys, J.C. Gates, B.J. Smith, P.G. Smith, I.A. Walmsley, Boson sampling on a photonic chip, Science 339 (6121) (2012) 798–801.
[54] Y. Subas, R.D. Somma, D. Orsucci, Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing, Phys. Rev. Lett. 122 (6) (2019) 060504.
[55] T. Toffoli, Reversible computing, in: Automata, Languages and Programming, Springer, Berlin Heidelberg, 1980, pp. 632–644.
[56] A. Tramontano, Protein Structure Prediction: Concepts and Applications, Wiley-VCH Verlag GmbH, 2006.
[57] M. Traykov, S. Angelov, N. Yanev, A new heuristic algorithm for protein folding in the HP model, J. Comput. Biol. 23 (8) (2016) 662–668.

[58] Jonathan Welch, Alex Bocharov, Krysta M. Svore, Efficient approximation of diagonal unitaries over the Clifford+T basis, Quantum Inf. Comput. 16 (1,2) (2016) 87–104.
[59] R. Wolfenden, Experimental measures of amino acid hydrophobicity and the topology of transmembrane and globular proteins, J. Gen. Physiol. 129 (5) (2007) 357–362.
[60] N. Yanev, M. Traykov, P. Milanov, B. Yurukov, Protein folding prediction in a cubic lattice in hydrophobic-polar model, J. Comput. Biol. 24 (5) (2017) 412–421.

Renata Wong has a PhD in Quantum Computing from the Department of Computer Science and Technology, Nanjing University, China. She is currently a postdoctoral researcher at the Physics Division, National Center for Theoretical Sciences in Taipei, Taiwan, R.O.C. Her research interests include quantum computation and information, foundations of physics, and linguistics. She also holds an M.A. in Sinology and a MSc. in Computer Science from Leipzig University, Germany.

Weng-Long Chang received the Ph.D. degree in Computer Science and Information Engineering from National Cheng Kung University, Taiwan, Republic of China, in 1999. He is currently Professor at the Department of Computer Science and Information Engineering in National Kaohsiung University of Science and Technology, Republic of China. His research interests include biological algorithms, quantum algorithms, quantum-molecular algorithms, Data Structures and Algorithms, and languages and compilers for parallel computing.