Scalable Guide Tree Construction Using Quantum Annealing for Multiple Sequence Alignment

Youngjun Park,[1,*] Juhyeon Kim,[2,*] and Joonsuk Huh[1,3,4,†]

[1] *Department of Chemistry, Sungkyunkwan University, Suwon 16419, Republic of Korea*
[2] *Department of Semiconductor Convergence Engineering, Sungkyunkwan University, Suwon 16419, Republic of Korea*
[3] *SKKU Advanced Institute of Nano Technology, Sungkyunkwan University, Suwon 16419, Republic of Korea*
[4] *Institute of Quantum Biophysics, Sungkyunkwan University, Suwon 16419, Republic of Korea*

A guide tree directs the sequence alignment order for multiple sequence alignment (MSA). Distance-based guide trees are widely used in MSA tools. However, many traditional algorithms for constructing these guide trees are unscalable and require a time complexity of $O(N^2)$ to $O(N^3)$, where N is the number of input sequences. In this study, we introduce a hierarchical agglomerative clustering (HAC) algorithm using quantum annealing that enables scalable guide tree construction. The proposed algorithm is based on two theoretical foundations: minimum evolution (ME) and molecular clock (MC). The ME requires finding the order of sequences that minimizes the total sum of branch lengths, which can be reduced to the traveling salesperson problem (TSP). The TSP is solved using quantum annealing. From the TSP solution, the algorithm can build a guide tree with a time complexity of $O(N)$ using the MC. In benchmark tests on BAliBASE 3.0 data set, the proposed guide trees outperform various classical HAC guide trees when aligning protein sequences with largely mutated termini. These guide trees will provide valuable insights into the study of protein termini.

I. INTRODUCTION

Multiple sequence alignment (MSA) is at the heart of interpreting genomic and proteomic data from various organisms. It plays a pivotal role in identifying *homologies* among biological sequences [1, 2]. Homology is essential for phylogenetics [3, 4] which has applications in such as disease treatment [5], pathogenesis [6], drug target identification [7], protein-protein interactions [8], enzyme research (e.g., Cas9 protein [9]), and the prediction of protein structure [10–12] and function [7].

Unfortunately, MSA is an NP-hard [13, 14] combinatorial optimization problem; computing an optimal alignment is intractable for large numbers of sequences. Therefore, MSA tools employ combinations of heuristic algorithms, including progressive alignment, iterative refinement, consistency-based schemes, profile hidden Markov models, and phylogeny-aware methods [1, 15–28]. Except for Dialign, all tools [1, 15–19, 21–28] perform *progressive alignment* that requires a *guide tree*.

Feng et al. [29] first proposed progressive alignment, in which the two most closely related sequences are aligned using dynamic programming [30, 31] such as the Needleman-Wunsch algorithm [32] and Smith-Waterman algorithm [33]. Then, the next closest sequence is aligned with the existing alignment. This process is repeated until all input sequences are incorporated into an MSA. For example, the simplest case of progressive alignment involves adding and aligning each sequence sequentially to the same end of the previous alignment. The alignment order is critical for ensuring the high quality of final output [34]. However, the number of possible alignment orders, even in this simplest case, is $O(N!)$, where N is the number of sequences.

A guide tree in progressive alignment is a rooted binary tree that determines the sequence alignment order [35]. Before the aligning process, guide tree construction algorithms build guide trees solely from distances between each pair of sequences. These distance-based algorithms should be distinguished from other phylogenetic inference methods, such as maximum-likelihood [36], parsimony [37], and Bayesian inference [38]. This is because guide tree construction algorithms require only distances, whereas these phylogenetic inference methods solve specific mathematical models independent of these distances.

Many guide tree construction methods utilize hierarchical agglomerative clustering (HAC) algorithms [39, 40], which are bottom-up heuristic approaches to building guide trees with low computational complexity. This is because finding an optimal tree is an NP-hard problem [41–44]. Neighbor-joining (NJ) [45] and unweighted pair group method with arithmetic mean (UPGMA) [46] are the most popular guide tree construction methods. These methods first compute a full distance matrix containing pairwise distances. Although this step can limit the scalability of these algorithms, parallel computing can effectively reduce the computation time as each matrix element can be independently calculated.

Extensive research [18, 24, 47–50] has been conducted to improve the time complexity of the NJ and UPGMA. However, the best implementations achieve a time complexity of $O(N^2)$ for a given distance matrix and are considered unscalable. PartTree [51] and a modified version of mBed [1, 52] are scalable methods with $O(N \log N)$ time complexity. However, these methods calculate a

* These authors contributed equally to this work.
† joonsukhuh@gmail.com

portion of a distance matrix and may yield less accurate trees. Moreover, PartTree is generally incompatible with other MSA tools [52].

Quantum annealing [53, 54] has garnered considerable attention for effectively solving combinatorial optimization problems. Many combinatorial optimization problems can be reduced to finding the ground state of a corresponding Ising Hamiltonian. The quantum adiabatic theorem and quantum tunneling facilitate obtaining the ground states of complex Ising Hamiltonians [55–58]. Quantum annealing can be performed on various platforms, including superconductors (D-Wave [59, 60]), 2D Rydberg atoms (QuEra [61] and Pasqal [62]) [63], Rydberg quantum wires [64], and nuclear magnetic resonance [65]. Lechner, Hauke, and Zoller proposed a scheme [66] that enables quantum annealing on various platforms. Moreover, quantum annealing has scaling advantages over simulated annealing [67] and path-integral Monte Carlo [68].

In this research, we present a scalable HAC algorithm that leverages quantum annealing to construct distance-based guide trees. This method is applicable to any MSA tool that employs distance-based guide trees. Our approach is based on two theoretical bases in guide tree construction: minimum evolution (ME) [69–72] and molecular clock (MC) [73–76]. The ME assumes that an optimal tree has a minimum total sum of branch lengths. Determining an order of sequences that yields a minimum total sum of branch lengths can be reduced to the traveling salesperson problem (TSP), which is an NP-hard problem [77, 78]. Here, the TSP is solved using quantum annealing. From the TSP solution, the algorithm constructs a guide tree with positive branch lengths in $O(N)$ time complexity using the MC.

The proposed algorithm can be implemented using universal quantum computation, as the quantum approximate optimization algorithm (QAOA) [79, 80] can also solve the TSP. In this study, we primarily focus on the scaling advantage of quantum annealing [67, 68].

Recent studies have reported quantum annealing and the QAOA approaches to MSA [81–83]. Nevertheless, their algorithms are limited to small toy problems and cannot insert gaps (as defined in Section II A) in the longest sequence. Previous research on MSA and tree construction based on the TSP [77, 84, 85] has been conducted. However, these studies did not specify how to determine and ensure positive branch lengths or even omitted assigning branch lengths on guide trees for the MSA process. These issues are addressed in the proposed algorithm (Sections II E and III).

The remainder of this paper is organized as follows. Section II introduces the key concepts used and establishes the main principle of this study. Section III presents the proposed guide tree construction algorithm. In Section IV, the protein MSA results from our guide trees are compared with those from classical HAC guide trees. The results revealed that the proposed guide trees outperformed numerous classical HAC guide trees when aligning protein sequences with significantly evolved terminal regions. Section V concludes the study. Additionally, appendices provide supplementary information about MAFFT (Appendix A), parallelizing distance matrix computation (Appendix B), D-Wave (Appendix C), and BAliBASE (Appendix D).

II. PRELIMINARIES

In this section, we review the basic concepts related to our distance-based guide tree construction algorithm. Sections II A and II B discuss pairwise alignment and the distance between two sequences, respectively, to explain the role of distance-based guide trees in MSA (Section II C). Section II D establishes the principle followed by the proposed algorithm (Principle 1), which combines the ME (Definition 5) and the MC (Definition 6). Finally, Section II E describes how the ME relates to the TSP. This TSP is solved using quantum annealing and the details of how the TSP solution is used in the proposed algorithm are presented in Section III.

A. Pairwise Alignments

Pairwise alignment is a process for aligning two sequences, a sequence with a group of aligned sequences, or groups of aligned sequences [30, 84, 86]. Sequences are defined as follows:

Definition 1. *A DNA, RNA, or protein sequence sequence s_i is a one-dimensional array of letters, each representing a nucleotide or amino acid: $s_i = s_i[1]s_i[2]\dots s_i[|s_i|]$, where $s_i[j]$ is the j-th letter in the sequence s_i, and $|s_i|$ is the length of s_i.*

Pairwise Alignment is a subroutine for progressive alignment and is related to defining the distance (Section II B) between two sequences. Algorithms such as the Needleman-Wunsch algorithm [32] and Smith-Waterman algorithm [33] are commonly used for sequence-to-sequence alignment. Group-to-group and sequence-to-group alignments are not discussed in detail here (see, for example, [87]).

Assume two DNA sequences that require alignment: $s_1 = \text{AAT}$ and $s_2 = \text{AAGT}$. Assuming these sequences are biologically related, they could have been identical in the past: AAT or AAGT. In the first case, a G was inserted into one of these sequences to become s_2. In the second case, a G was deleted from one of these sequences to become s_1. The next step is to mark the location in the sequences where the insertion or deletion likely occurred. For this purpose, a *gap* character (or simply, gap) '$-$' is inserted to represent the insertion or deletion of a nucleotide (or amino acid) in sequences.

In this example, it is clear that a gap should be inserted into AAT to represent the insertion (the first case) and deletion (the second case). When trying to

insert a gap into AAT, four *gap-inserted sequences* are possible: −AAT, A−AT, AA−T, and AAT−. This indicates that AA−T is the ideal choice here. The results of pairwise alignment are not always optimal: (−AAT, AAGT), (A−AT, AAGT), (AA−T, AAGT), and (AAT−, AAGT). (AA−T, AAGT) is the *optimal pairwise alignment* among them.

The gap-inserted sequence, pairwise alignment, and optimal pairwise alignment are defined as follows:

Definition 2. *A gap-inserted sequence g_i comprises the original characters from s_i interspersed with gaps and maintains the original order of characters in s_i.*

Definition 3. *A pairwise alignment of two different sequences s_i and s_j ($|s_i| \geq |s_j|$) produces either:*

1. *two gap-inserted sequences g_i and g_j of the same length ($|g_i| = |g_j|$), or*
2. *the unchanged sequence s_i and a gap-inserted sequence g_j, such that $|s_i| = |g_j|$.*

Although we did not show an example of the first case in Definition 3, gaps can generally be inserted into both sequences.

Definition 4. *The optimal pairwise alignment of given sequences s_i and s_j is a pairwise alignment of the maximum matching (or score).*

The calculation methods for *score* in Definition 4 vary considerably and are difficult to generalize. Therefore, a detailed discussion is beyond the scope of this research. To provide some insight into these methods, consider the previous DNA pairwise alignment with the following scores: match (+5), mismatch (−2), and gap (−6) [30]. For the alignments (−AAT, AAGT), (A−AT, AAGT), and (AAT−, AAGT), there are two matches, one gap, and one mismatch, respectively; each of these results in a score of +2. The alignment (AA−T, AAGT) has three matches and one gap, resulting in a score of +9. Therefore, the optimal pairwise alignment (AA−T, AAGT) has the highest score among all possible pairwise alignments. For protein alignment, matching and mismatching scores are calculated using substitution matrices such as PAM [88] and BLOSUM [22].

B. Distance Between Two Sequences

The distance between two sequences indicates how closely or distantly related they are [43]. Distance-based guide trees are constructed using the distances between each pair of input sequences. If the distance is short, the two sequences are closely related; otherwise, they are in a distant relation.

Two types of distance measures are employed: approximate measures (e.g., k-mer, k-tuple, and Wu-Manber algorithm [22, 24, 89–91]) and exact measures (e.g., dynamic programming [22, 89, 90]). For sequences of length L, exact measures have a quadratic time complexity ($O(L^2)$); however, this approach can be time-consuming for long sequences. Therefore, large-scale MSAs use approximate measures, which typically grow in a linear time complexity ($O(L)$).

Consider the following example. In Section IV, benchmark tests were conducted based on distances evaluated by MAFFT L-INS-i [22, 90] as follows:

$$d(s_i, s_j) = \left\{ 1 - \frac{S(s_i, s_j)}{\min(S(s_i, s_i), S(s_j, s_j))} \right\} / f(x, y), \quad (1)$$

where $d(s_i, s_j)$ is the distance between sequences s_i and s_j, and $f(x, y)$ is a function of x and y. Here, x and y represent the lengths of the longer and shorter sequences, respectively, and $f(x, y)$ includes empirically determined constants. The score $S(s_i, s_j)$ of sequences s_i and s_j is calculated differently across various MAFFT algorithms. L-INS-i, which aims for an accurate alignment, computes the score using an exact measure. In Appendix A, we provide an MAFFT command for obtaining a distance matrix.

C. Distance-Based Guide Trees in MSAs

A distance-based guide tree is a binary, undirected, acyclic, and rooted tree constructed using a distance matrix, where each leaf node represents an input sequence [77, 92]. It directs the order in which input sequences are aligned in progressive alignment [35, 92]. An internal node, referred to as a *clade*, represents an unknown common ancestor, and the root node represents the unknown common ancestor of all input sequences. For each branch on a guide tree, the branch length is a real positive number that indicates how closely two nodes are related. For example, a short branch length is assigned when two nodes are closely related. The branch length can also be interpreted as a real positive value proportional to the number of evolutionary events along that branch. The branch length is determined by the distance matrix and a specific guide tree construction algorithm. In Section III, we demonstrate how the proposed algorithm determines branch lengths.

Fig. 1 shows two example guide trees for four RNA sequences, s_1 (GGCCAA), s_2 (GGCCU), s_3 (CUCUA), and s_4 (UUGGGGUU) with arbitrarily determined branch lengths and tree topologies. Fig. 1(a) illustrates an example MSA process. First, s_1 and s_2 are aligned (a sequence-to-sequence alignment). Then, the alignment of s_1 and s_2 is aligned with s_3 (a group-to-sequence alignment). Finally, the alignment of s_1, s_2, and s_3 is aligned with s_4, generating an MSA. In Fig. 1(b), s_1 and s_2 are aligned separately to s_3 and s_4, and the two pairwise alignments are then aligned to produce an MSA (a group-to-group alignment). As shown in Fig. 1, different guide trees yield different MSA outcomes. Note that a

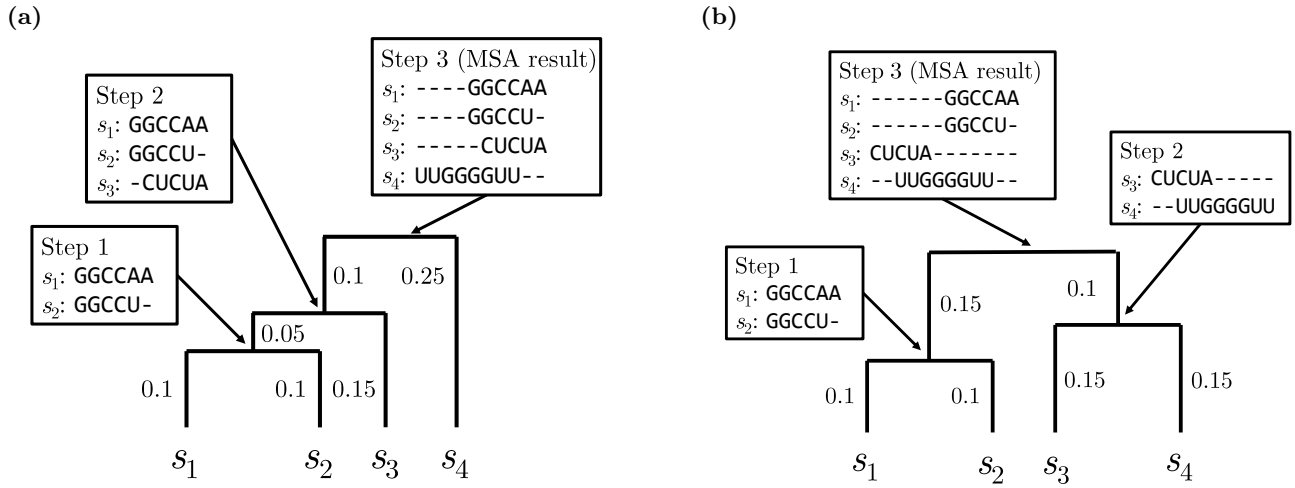

FIG. 1. Two example guide trees with arbitrarily chosen tree topologies and branch lengths. A number near a branch indicates its branch length. Four sequences (s_1, s_2, s_3, and s_4) are some arbitrary RNA sequences. We referred to Fig. 1 in [92] and created two figures of different example situations. The MSA results are provided by MAFFT FFT-NS-1 (a progressive alignment with no iterations) [22, 90].

guide tree has a hierarchical structure and is typically constructed using a HAC algorithm.

MSA procedure is affected by branch lengths; for instance, ClustalW leverages branch lengths to postpone the alignment of highly divergent sequences until after the alignment of all other sequences is complete [52]. More accurate branch lengths on guide trees yield better MSA results [93]. However, when a true phylogenetic tree is unknown, assessing the accuracy of branch lengths remains challenging. Therefore, in many cases, the accuracy of branch lengths and tree topology is indirectly evaluated based on an MSA result [93, 94]. Consequently, Section IV presents the MSA scores to compare the accuracy of traditional guide trees with the one developed here.

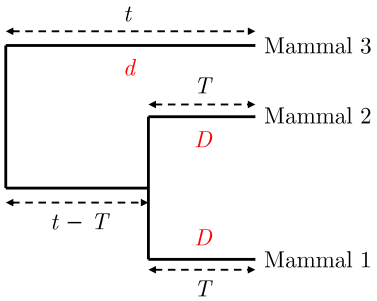

FIG. 2. The MC assumes that the rate of molecular evolution is approximately constant across different species. t and T: the time of divergences, and d and D: the number of amino acid differences in a specific protein.

D. ME and MC: Theoretical Foundations in Guide Trees

The ME and MC are two popular theoretical foundations in phylogenetic research and guide tree construction. We discuss and define these concepts (Definitions 5 and 6) and explain how they are combined in the proposed algorithm (Principle 1).

The concept of ME is that an optimal tree has a minimum number of evolutionary events [45, 69, 70]. Because each branch length is proportional to the number of evolutionary events (Section II C), an optimal tree has the minimum total sum of branch lengths, as per the ME.

The term ME was first introduced by Cavalli-Sforza et al. in the 1960s [72]. In the 1980s, Saitou et al. proposed the NJ algorithm that uses ME as a theoretical foundation [45]. The NJ tree uses the ordinary least-squares (OLS) for estimating branch lengths, as discussed by Rzhetsky et al. [70]. In the 2000s, Gascuel et al. [71] discovered that the ME criterion used by Saitou et al. was the balanced minimum evolution.

The ME is defined as follows [45, 69, 70]:

Definition 5. *The ME assumes that the optimal tree for a given set of sequences is most likely the one with the minimum number of evolutionary changes, such that it has the minimum total sum of branch lengths.*

The following two points about the ME must be clarified:

- In this work, branch lengths are not measured using the OLS method.
- Following Definition 5, the concept of ME was used to minimize the total sum of branch lengths using a given distance matrix (Section II E).

The MC is an influential concept in evolutionary studies and was first introduced by Zuckerkandl and Pauling

[75, 76] for estimating the divergence time of different mammals. Referring to their original work and a review paper [74], we explain the concept using a slightly different example (Fig. 2). Assume that a fossil record of mammals 1 and 2 indicates that they diverged four million years (Myr) ago ($T = 4$ Myr). By comparing specific proteins in present organisms, such as α-hemoglobin proteins, of mammals 1 and 2, the number of amino acid differences is two substitutions ($D = 2$ amino-acids substitution). They used the MC for calculating the time of divergence (t) of mammals 1 and 3; it is defined as follows:

Definition 6. *The MC assumes that the rate of molecular evolution is approximately constant in different species.*

If the number of amino acid differences between mammals 1 and 3 is three substitutions ($d = 3$ amino-acids substitution), the following relationship can be deduced: $T : D = t : d$. This is because the molecular evolution is assumed to be constant. Then, $t = d \cdot \frac{T}{D} = 6$ Myr.

Although the accuracy of the MC has been debated, it has formed the basis for numerous studies [74]. In the absence of fossil or biogeographical records, the MC provides a reasonable method for inferring the time of divergence. Therefore, the MC is particularly relevant in constructing guide trees for MSAs, where guide trees are created even without comprehensive biological data. Consequently, the MC remains a valid and practical assumption for guide tree construction.

Most importantly, the MC assumes the distances from the root to each leaf node to be the same (Fig. 2).

In this research, the concepts of ME (Definition 5) and MC (Definition 6) are combined as stated in the following principle. This combination is referred to as the minimum evolution with the molecular clock (MEMC) principle.

Principle 1. *(MEMC principle) In the optimal guide tree for a given set of sequences, (1) the total sum of branch lengths is minimized (the ME), (2) and the distances from the root to each leaf node are equal (the MC).*

In the proposed algorithm (Section III), the ME is first applied and followed by the MC to construct a guide tree. As the ME is related to the TSP, it will be further discussed in Section II E before introducing the algorithm in Section III.

E. The ME and the TSP

There are two challenges in minimizing the total sum of branch lengths. (1) We lack information about common ancestors (clades); therefore, their sequences cannot be confidently inferred. (2) The number of possible tree topologies is extremely large. For N leaf nodes, there are $1 \cdot 3 \cdot 5 \cdot \cdots \cdot (2N-3) = \prod_{k=2}^{N}(2k-3) = \frac{(2N-3)!}{2^{N-2}(N-2)!}$ possible tree topologies [95]. Thus, the total sum of branch lengths must be computed solely from a given distance matrix of input sequences, and a brute force algorithm is impractical.

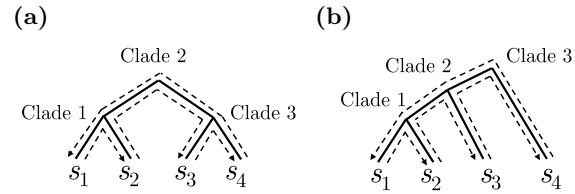

FIG. 3. The CS method can calculate the total sum of branch lengths independent of tree topologies.

Korostensky et al. [77, 84] proposed the circular sum (CS) method to find the *circular order* of input sequences, which can be used to construct a tree with the minimum total sum of branch lengths. A circular order C of input sequences is defined here as follows:

Definition 7. *A circular order C of input sequences is an order in which each sequence s_i appears exactly once.*

A given set of N sequences has $(N-1)!$ possible circular orders. For a given set of four sequences (s_1, s_2, s_3, and s_4), an example of a circular order is $s_3 \to s_1 \to s_4 \to s_2$. This order is circular because the order can restart from the first sequence after the fourth sequence. In this case, $C_5 = C_1$, which means the sequence in the fifth position of C is the same as that in the first position (s_3). For simplicity, the circular order is represented as a single loop: $C = [s_3, s_1, s_4, s_2]$.

The CS [77, 84] is the sum of distances between each pair of sequences that are consecutive in a given circular order C. Korostensky et al. used optimal pairwise alignment scores to compute the CS and assign branch lengths to a tree. However, these optimal pairwise scores can be negative for sequences with low similarity, and negative branch lengths are incompatible with recent MSA tools. In the present work, we use positive distances computed by an MSA tool, where we intend to input our guide trees.

Assume four sequences and a circular order represented as $C = [s_1, s_2, s_3, s_4]$. Then, the CS of distances in C is

$$\begin{aligned} &d(C_1, C_2) + d(C_2, C_3) + d(C_3, C_4) + d(C_4, C_5) \\ &= d(s_1, s_2) + d(s_2, s_3) + d(s_3, s_4) + d(s_4, s_1), \end{aligned}$$

where $C_5 = C_1$. As shown in Figs. 3(a) and 3(b), the CS method adds each branch length exactly twice to the total sum for any tree topology of the same C. Thus, by dividing the above sum by 2, the total sum of branch lengths can be obtained before determining the tree topology.

Therefore, a circular order C that minimizes the fol-

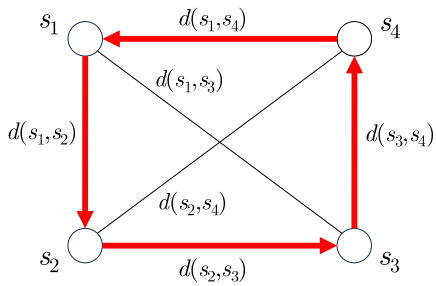

FIG. 4. MECO problem (Problem 1) with four input sequences (s_1, s_2, s_3, and s_4). An example solution is $C = [s_1, s_2, s_3, s_4]$. Choosing a starting point is unimportant as the tour is circular.

lowing sum is directly related to the ME (Definition 5):

$$\frac{1}{2}\sum_{n=1}^{N} d(C_n, C_{n+1}), \tag{2}$$

where $d(C_n, C_{n+1})$ is the distance between the sequences in the n-th and $(n+1)$-th positions in the C. If MAFFT is used, the $d(C_n, C_{n+1})$ can be computed using Eq. 1.

Korostensky et al. [77, 84] discovered that the problem of finding the circular order C that minimizes Eq. 2 can be reduced to the TSP. The TSP is defined as follows:

Definition 8. *The TSP is a problem of finding the shortest path in a given graph that visits each node once at each step and returns to the starting point.*

The minimum evolution circular order (MECO) problem can be stated as follows (Fig. 4):

Problem 1. *(MECO problem) Find a circular order C (Definition 7) that minimizes Eq. 2.*

Fig. 4 shows an example solution to the MECO problem with four sequences. The MECO C serves as a starting point for our algorithm.

III. METHODS

Our guide tree construction algorithm is scalable by leveraging quantum annealing and is based on the MEMC principle (Principle 1). A distance matrix is assumed to be given or can be obtained using parallel computing (Appendix B). The first phase of the algorithm involves solving the MECO problem (Problem 1) using quantum annealers. The solution is an MECO, a circular order C. The second phase involves determining a tree topology and positive branch lengths from the MECO C, with a time complexity of $O(N)$. This algorithm addresses drawbacks in the work of Korostensky et al. [77] and is inspired by UPGMA [46]. Section II explains the terms and concepts used here. Figure 5 illustrates the overall algorithm with a simple example.

A. MEMC Guide Tree Algorithm

1. *Generating a TSP Graph for Finding the MECO*

A distance matrix, which is either provided or obtained through parallel computing (Appendix B), is first mapped to a TSP graph to solve the MECO problem (Problem 1). Assume N input sequences. Each node in the graph corresponds to an input sequence s_i. The weight of the edge between two nodes representing s_i and s_j is given by the distance $d(s_i, s_j)$ (Section II B). The TSP graph is fully connected, undirected, and satisfies $d(s_i, s_j) = d(s_j, s_i)$ for all $i \neq j$ [97]. An example is shown in Step 1 of Fig. 5.

Note that these distances must be calculated using the same MSA tool into which the guide trees will be input, as using different distance measures can lead to misinterpretations regarding the evolutionary closeness between input sequences. Suppose MSA tool A estimates distances as smaller than those measured by MSA tool B. Constructing a guide tree with distances from MSA tool A and inputting it into MSA tool B may cause the latter to misinterpret the closeness of sequences. Thus, MSA tool B may align them as if they are closer than they actually are, leading to a low-quality MSA result.

2. *Finding the MECO of the Graph Using a Quantum Annealer*

A quantum annealer is used to find the TSP order in the constructed graph. This TSP order is equivalent to the MECO. The problem must be formulated as a binary model to solve the TSP on a quantum annealer. We map each spin of a qubit, $\sigma \in \{-1, +1\}$, to a binary value b using the formula $(\sigma + 1)/2$. A binary variable $b_{i,p}$ is defined to take the value of 1 if the node representing s_i is selected at the p-th step of a circular order C, and 0 otherwise:

$$b_{i,p} = \begin{cases} 1 \ (s_i = C_p), \\ 0 \ (s_i \neq C_p). \end{cases} \tag{3}$$

Then, the TSP can be formulated as the following Ising model [97] using the binary variables defined in Eq. 3:

$$H = A\sum_{i=1}^{N}\left(1 - \sum_{p=1}^{N} b_{i,p}\right)^2 + A\sum_{p=1}^{N}\left(1 - \sum_{i=1}^{N} b_{i,p}\right)^2 + B\sum_{\substack{i,j=1,\\ i\neq j}}^{N} d(s_i, s_j)\sum_{p=1}^{N} b_{i,p}b_{j,p+1}, \tag{4}$$

where A is a positive constant, $0 < B\cdot\max(d(s_i, s_j)) < A$, and $d(s_i, s_j)$ is the distance between the sequences s_i and s_j. The node selected at the $(N+1)$-th step is the same as the initial node. As $B \cdot \max(d(s_i, s_j)) < A$,

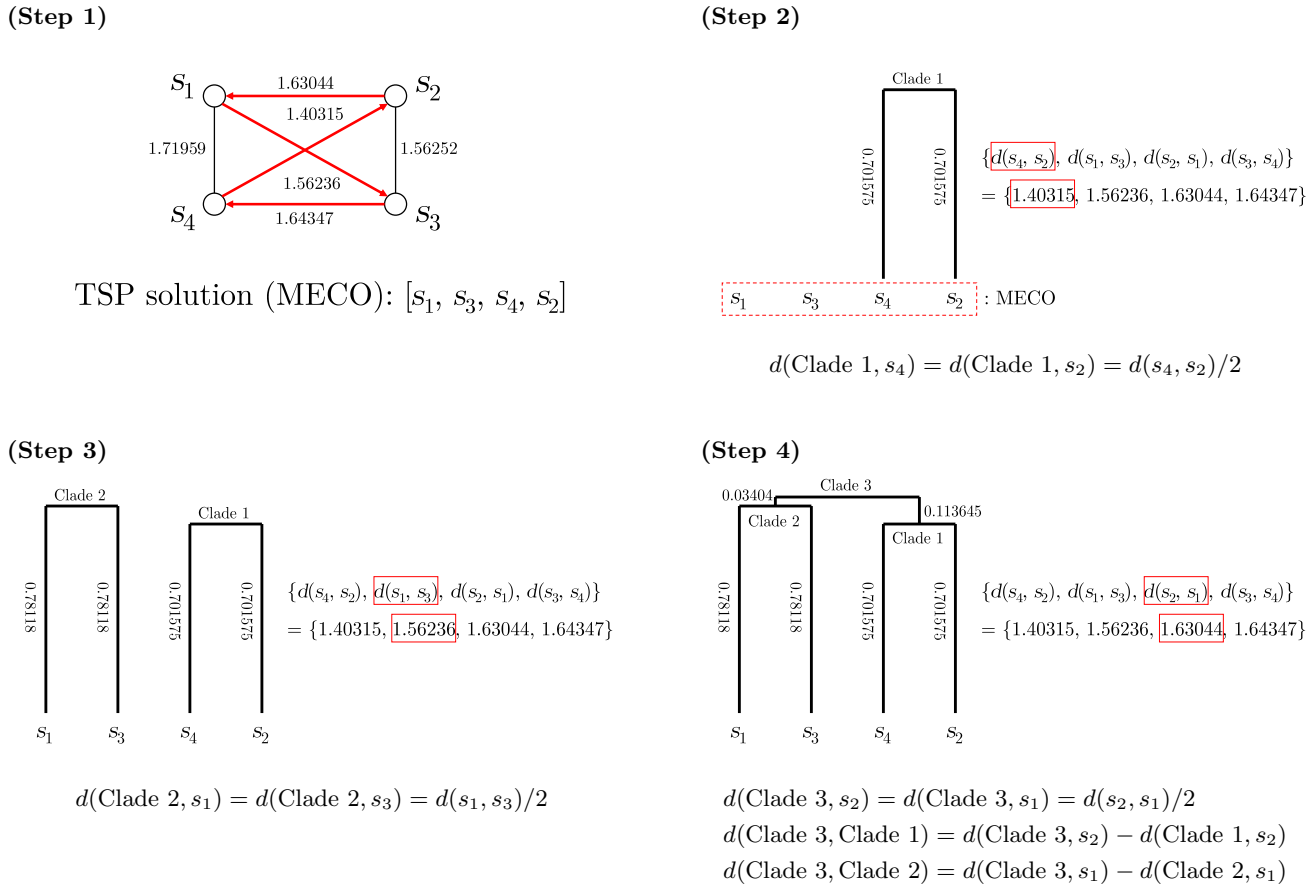

FIG. 5. MEMC guide tree construction algorithm. A simple example (BB11001 in BAliBASE 3.0 [96]) is illustrated. Step 1 is the first phase, and Steps 2–4 are the second phase. In Step 1, the distances on the graph are calculated using MAFFT L-INS-i. Step 1 finds the TSP order in the graph, which is equivalent to the MECO (Problem 1). Using the MECO, the distances between consecutive sequences in the circular order are saved and sorted. The tree topology and positive branch lengths are determined following Steps 2–4 while retaining the MECO. At each step, the two closest leaf nodes or clades are connected, and the newly generated branch lengths are determined. This step is repeated $N-1$ times for N input sequences until a tree is constructed. When two unconnected leaf nodes s_i and s_j are connected to a new clade, the branch length between the clade and each leaf node is half of $d(s_i, s_j)$. Note that the distance matrix is not updated after each clustering. Step 4 involves determining the branch length between a new clade and an existing clade.

non-Hamiltonian cycles are strictly excluded as solutions to the problem. Quantum annealers can obtain the approximate ground state of the Ising Hamiltonian corresponding to the Ising model (Eq. 4) with advantages in scalability [67, 68]. Therefore, using quantum annealing in the proposed algorithm enables scalable guide tree construction. Additionally, the QAOA can be utilized to find the approximate ground state of the same Ising model, which enables universal quantum computation for the proposed algorithm [79, 80]. The ground state configuration of the qubits for the Ising model corresponds to the binary variables. For instance, if $b_{1,1}$, $b_{3,2}$, $b_{4,3}$, and $b_{2,4}$ are all ones and the others are all zeros, then the MECO is $[s_1, s_3, s_4, s_2]$ (Step 1 in Fig. 5).

3. *Determining the Tree Topology and Branch Lengths*

After finding the MECO, the topology and branch lengths of the guide tree are determined using the concept of MC (Definition 6). First, each leaf node is assigned to represent an input sequence in the order of the MECO C. Second, the N distances between consecutive pairs of sequences in C are saved as a set $D = \{d(C_i, C_{i+1})\}_{i=1}^{N}$. The set D can be sorted in increasing order of distance using radix sort, which requires a worst-case time complexity of $O(kN)$ [98]. This sorting is necessary to ensure positive branch lengths on the tree. The number of significant digits k in the distances is consistent, and typically six to seven digits are sufficient for constructing guide trees. Therefore, $k \ll N$ in practical cases. Finally, the nodes or clades are found and connected, and branch lengths are assigned in increasing order of distance, as specified by the sorted set D.

This algorithm is demonstrated in Fig. 5, an example of four input sequences. Suppose a quantum annealer provided the MECO as $C = [s_1, s_3, s_4, s_2]$; $D = \{d(s_1, s_3), d(s_3, s_4), d(s_4, s_2), d(s_2, s_1)\}$. A sorted set D, for example, is $\{d(s_4, s_2), d(s_1, s_3), d(s_2, s_1), d(s_3, s_4)\}$.

From the sorted set D, the minimum distance is identified, and the corresponding two leaf nodes are connected to a new clade (Step 2 in Fig. 5). Here, the minimum is $d(s_4, s_2)$, so s_4 and s_2 are connected to a new clade (Clade 1). This new clade is assumed to be equidistant from the connected leaf nodes. This assumption is widely accepted in many guide tree construction algorithms such as UPGMA [99, 100]. Thus, the branch lengths are set as $d(\text{Clade } 1, s_4) = d(\text{Clade } 1, s_2) = d(s_4, s_2)/2$.

After connecting either nodes or clades, classical HAC algorithms update distances to reflect new relationships between the new clade and other existing nodes and clades. However, in our algorithm, distances are not updated because the MECO obtained using quantum annealing is a good approximation to the global minimum of the total sum of branch lengths. To update the distances, a new TSP will have to be solved to find the MECO for the updated distance matrix. This requires solving a total of $O(N)$ TSPs, making the algorithm inefficient. Therefore, we decided not to update distances, and in Section IV, we will demonstrate that this approach works well for specific cases.

The process is repeated without updating the distances in the sorted set D until all leaf nodes are connected to a single tree (Steps 3 and 4 in Fig. 5). Note how branch lengths are computed when two clades are connected to a new clade (Step 4 in Fig. 5). The branch length between the new clade and one of the connected clades is the difference between the branch lengths of these clades. Thus, $d(\text{Clade } 3, \text{Clade } 1) = d(\text{Clade } 3, s_2) - d(\text{Clade } 1, s_2)$, and $d(\text{Clade } 3, \text{Clade } 2) = d(\text{Clade } 3, s_1) - d(\text{Clade } 2, s_1)$, where $d(\text{Clade } 3, s_2) = d(\text{Clade } 3, s_1) = d(s_2, s_1)/2$. This computation guarantees positive branch lengths because $d(s_2, s_1) > d(s_1, s_3) > d(s_4, s_2)$ is ensured by the sorted distance set D. A similar computation is applied when a clade and a leaf node are connected to a new clade.

To ensure efficient implementation of the algorithm, a disjoint-set data structure combined with two optimization techniques, namely union by rank and path compression, is employed [98]. The process begins by creating a disjoint-set data structure for N leaf nodes, which has a time complexity of $O(N)$. Path compression accelerates the identification of the clade to which a leaf node belongs, and union by rank efficiently joins two subtrees. Both operations have a time complexity of $O(\alpha(N))$, where $\alpha(N)$ is a very slowly growing function known as the inverse Ackermann function. Our algorithm executes a constant $O(1)$ number of these operations at each step, leading to a total time complexity of $O(N\alpha(N))$ over $O(N)$ steps. Note that $\alpha(N) \leq 4$ for any practical value of N [98]. In conclusion, the worst-case time complexity of the proposed algorithm, including the sorting step, can be considered linear: $O(N(k + \alpha(N))) \approx O(N)$.

IV. RESULTS AND DISCUSSION

In this section, we investigate the impact of the proposed MEMC guide trees (Section III) on MSA results using MAFFT L-INS-i [22], an iterative, progressive, and consistency-based alignment method. However, our guide tree construction method is also compatible with other MSA tools and is not limited to MAFFT. Protein MSA benchmark tests were conducted using BAliBASE 3.0 [96] to compare the performance of our guide trees with classical distance-based HAC guide trees. A D-Wave quantum-classical hybrid solver was used to simulate quantum annealing for constructing the MEMC guide trees. Appendix C includes the details of the D-Wave implementation. Table I summarizes the average total column (TC) scores from the benchmark tests.

A. BAliBASE 3.0: Protein MSA Benchmark

BAliBASE 3.0 is a widely used protein MSA benchmark [96] for evaluating the performance of MSA tools [1, 24, 28, 102]. It contains six different test categories, each with dozens of test cases to be aligned: BB1-1, BB1-2, BB2, BB3, BB4, and BB5.

Core blocks (Appendix D) are considered for calculating benchmark scores. Core blocks represent reliably aligned columns with no gaps in the MSA results [96]. They are determined based on whether three-dimensional structures are superposable and sequences are conserved at these sites.

The sum of pairs (SP) and TC scores are criteria used to assess MSA results [1, 96]. The SP score is calculated by counting the number of pairwise matches to core blocks in a reference. The TC score is computed by counting the number of columns that exactly match the core blocks in a reference. These scores range from 0 to 1, with higher scores indicating better alignments. As the TC is a stricter criterion than the SP, only TC scores are shown in Table I.

B. Performance Evaluation: MEMC vs. Classical Guide Trees

Here, we evaluate the performance of the MEMC guide trees against classical distance-based HAC guide trees, including complete linkage, flexible, modified UPGMA, single linkage, UPGMA, UPGMC, Ward, WPGMA, and WPGMC [100]. The modified UPGMA is the default guide tree in MAFFT [103]. These classical guide trees are based on the MC, ensuring that the root-to-leaf distances are uniform. In contrast, the NJ is not based on the MC and uses the OLS to estimate branch lengths (Section II D). Due to the algorithmic similarities, we

Guide Tree	BB1-1 (38 families)	BB1-2 (44 families)	BB2 (41 families)	BB3 (30 families)	BB4 (49 families)	BB5 (16 families)	Total (218 families)
MEMC	0.4381	0.8541	0.4240	0.5517	**0.6481**	0.5882	0.5933
Complete linkage	0.4405	0.8511	0.4748	0.6543	0.5963	0.6093	0.6067
Flexible	**0.4532**	0.8536	0.4555	0.6007	0.6408	0.6088	0.6083
Modified UPGMA (default)	0.4444	0.8486	0.4854	0.6559	0.6295	0.6207	0.6173
Single linkage	0.4399	0.8497	0.4779	0.6573	0.6302	**0.6278**	0.6162
UPGMA	0.4469	**0.8584**	0.4758	**0.6676**	0.6386	0.6127	**0.6210**
UPGMC	0.4433	0.8479	0.4560	0.6286	0.6062	0.6084	0.6016
Ward	0.4488	0.8484	**0.4861**	0.6653	0.6130	0.6164	0.6155
WPGMA	0.4529	0.8549	0.4633	0.6632	0.6257	0.6055	0.6150
WPGMC	0.4531	0.8507	0.4549	0.6168	0.6188	0.6101	0.6050

TABLE I. Average TC scores from BAliBASE 3.0 benchmark tests. All guide trees were constructed using the same distance matrices. We used MAFFT (ver. 7.526) L-INS-i for MSAs. Appendix A provides the details of the commands used to obtain the distance matrices and the MSAs. The second row shows the MSA results when the MEMC guide trees (Section III) were used. From the third row onward, the MSA results using classical guide tree algorithms are shown in alphabetical order. We set the minimum branch length to 0.00001 to prevent negative branch lengths in these classical guide trees. Modified UPGMA is the default guide tree in MAFFT. The term 'flexible' denotes the strategy introduced by Lance et al. [101], with its β value arbitrarily set to 0.5. For each category, the best method is emboldened. UPGMA achieved the highest average TC score over 218 cases. However, in BB4, the MEMC guide tree outperformed all classical methods, including UPGMA.

have focused our performance comparisons on them and excluded the NJ from our analysis.

Table I shows the overall results. We used default input parameters and set the maximum number of iterations to two for MAFFT L-INS-i (Appendix A). Iterations were included because they are always needed to improve MSA results in practical applications. However, we limited them to two to preserve the influence of the initial guide trees. On average, UPGMA guide trees achieved higher TC scores than other methods. However, no single algorithm consistently outperformed the others across all categories. Notably, the MEMC guide trees performed the best in BB4.

In BB1-1 and BB1-2, the score differences between the best and worst methods were the smallest among different categories. The average numbers of input sequences per case were 6.87 and 9.0 for BB1-1 and BB1-2, respectively, which are considerably smaller than those in other categories. For these small-scale cases, the guide trees did not exhibit significant differences in topologies and branch lengths (Fig. 6(a) and 6(b)).

In BB2 and BB3, the MEMC guide trees demonstrated comparatively lower performance compared with other categories. These categories comprise sequences of similar lengths, which are clustered into several subfamilies [96, 104]. A subfamily is a subset of input sequences that have closer biological relationships (>25% sequence identity) than other sequences. Consequently, branch lengths within a subfamily are short, whereas branch lengths between subfamilies are long. Classical trees, such as those produced by UPGMA, typically have a structure where the clades near the root are not densely populated (Fig. 6(d)). In contrast, the proposed method tends to produce trees where the clades adjacent to the root are densely populated (Fig. 6(c)). This is because the MECO can involve long distances, whereas classical HAC algorithms choose the shortest distance in each updated distance matrix. However, this algorithmic feature was beneficial to the proposed algorithm in BB4 test cases.

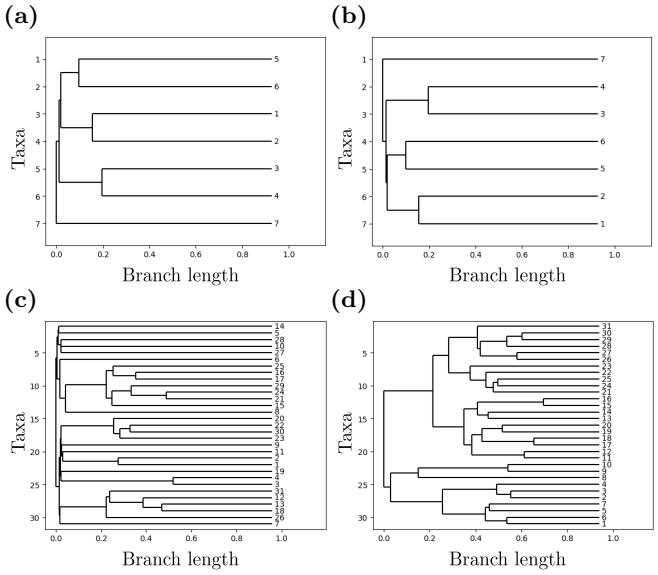

FIG. 6. Examples of guide trees. For simplicity, only several representative cases are presented: (a) MEMC guide tree (BB11027), (b) Modified UPGMA (BB11027), (c) MEMC guide tree (BB30002), and (d) UPGMA (BB30002).

In BB4, the MEMC guide trees outperformed all classical trees. In BB5, the default and single linkage guide trees exhibited the best performance. Test cases in categories BB4 and BB5 comprise sequences of varying lengths; some sequences are very long, and others are relatively short [96, 104]. After performing MSA, large external and internal gaps are inserted into BB4 and BB5 test cases, respectively. Note that a point mutation can introduce a large external gap, whereas introducing a large internal gap requires numerous point mutations.

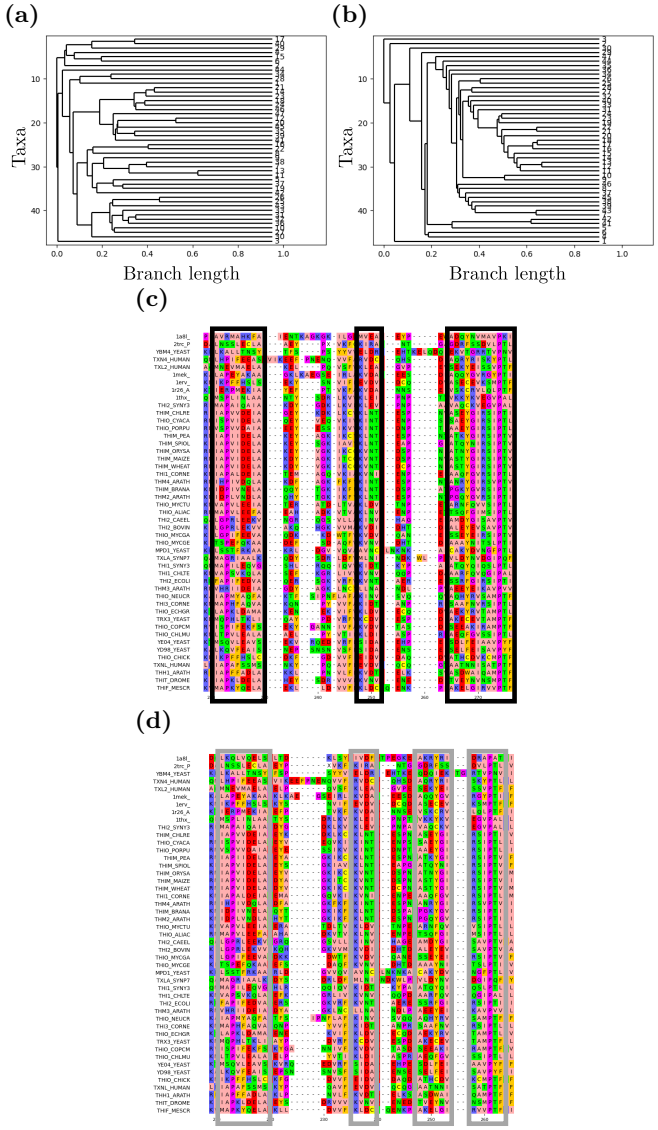

FIG. 7. Examples of guide trees and MSAs. (a) MEMC guide tree (BB40044). (b) Modified UPGMA (BB40044). (c) A portion of the MSA result obtained using the MEMC guide tree. All core blocks in the reference are perfectly matched to blocks in (c), as indicated by dark squares, resulting in a TC score of 1. (d) A portion of the MSA result using the modified UPGMA guide tree. Gray squares denote blocks related to the core blocks but not exactly matching them, leading to a TC score of 0.

This is because protein synthesis is extended if a point mutation changes a stop codon to a sense codon. In contrast, protein synthesis is prematurely terminated if a point mutation converts a sense codon to a stop codon. If these resulting proteins have been naturally selected, new species can emerge and form new bifurcations in an evolutionary tree. However, a point mutation that changes one amino acid to another generally affects only that amino acid.

As a few point mutations can result in a large external gap, many early bifurcations likely occurred in the BB4 test cases. As shown in Fig. 7(a), the proposed guide trees have short branch lengths near the root, indicating these early bifurcations. The MSA process reflected this feature and provided a perfect match to the core blocks in the reference (Fig. 7(c)). In contrast, the modified UPGMA guide trees did not capture these early bifurcations well (Fig. 7(b)). As a result, this guide tree achieved a TC score of 0 (Fig. 7(d)).

Aligning multiple sequences similar to the BB4 cases is particularly important in biological research. The ends of a protein, known as the N/C termini, play significant roles in molecular signaling between proteins and molecules [105]. Various post-translational modifications at these termini modulate protein activities and functions [106]. Moreover, abnormalities in the modification mechanism can induce certain diseases such as cancers [106, 107]. However, structural analyses alone cannot provide sufficient insights into the role of N/C termini, as intrinsically disordered protein termini do not have fixed structures [108–111]. Thus, using guide trees built from the proposed algorithm can enhance the quality of MSAs, which further facilitates a better understanding of the N/C termini in proteins.

It is well known that the accuracy of classical HAC guide trees, including scalable ones, rapidly decreases as the problem size increases (Fig. 3 in Ref. [94]). Although scalable guide tree algorithms such as PartTree are not recommended for small problems [112], the proposed algorithm showed comparable results with more accurate methods in small-scale problems. We anticipate that the proposed algorithm may surpass various classical guide tree construction algorithms in large-scale problems.

V. CONCLUSION

This study introduces a scalable HAC algorithm for constructing distance-based guide trees. We established and used the MEMC principle (Principle 1) to find an order of sequences that minimizes the total sum of branch lengths and to construct a guide tree from this order. Unlike traditional research based on the TSP, the proposed algorithm is scalable and produces guide trees with positive branch lengths.

This scalability is provided by two aspects of the proposed method. (1) The MECO problem (Problem 1), which is introduced by the ME, is solved using quantum annealing. Note that this problem can also be solved using the QAOA implemented in a circuit-based quantum computer. (2) The MECO and the MC allow for the construction of a guide tree with linear time complexity. The MECO serves as a good starting point for the algorithm and eliminates the need to update the distance matrix. These are this paper's main contributions, enabling scalable guide tree construction. Moreover, the proposed approach is a practical quantum annealing algorithm, whereas recent quantum computing algorithms

for MSA cannot solve real-world problems.

BAliBASE 3.0, a protein MSA benchmark, demonstrated that guide trees generated by the proposed method improved the quality of MSA results when input sequences involved large external gaps (BB4). Although this method has achieved this improvement, further work is needed to enhance the TC scores in other categories. Note that the BB4 test cases are heavily related to mutations in protein N/C termini, which play a critical role in protein signaling and post-translational modifications. Moreover, the structural analyses of N/C termini are challenging because many proteins have intrinsically disordered tails. Thus, improving the quality of MSAs using guide trees constructed from the proposed algorithm can considerably benefit N/C termini research.

The proposed algorithm is not confined to protein MSA; therefore, further large-scale MSA benchmark tests, including DNA and RNA, are required for a more comprehensive understanding of the algorithm. These tests will be possible in the near future as the number of qubits in quantum annealers has been increasing.

An interesting perspective on our guide tree algorithm is its applicability to general clustering problems. For instance, chemical similarity research [113] employs HAC such as Ward [114] to discover structure-activity relationships of chemicals based on their chemical structures. The proposed guide tree algorithm may be useful in clustering chemicals that share a common structure but have diverse additional functional groups, such as corticosteroid derivatives.

VI. CODE AVAILABILITY

The source code is provided via a GitHub repository (`https://github.com/Bevacizumab/memcguidetree`).

ACKNOWLEDGMENTS

Our thanks to Gwonhak Lee, Jiyun Lee, Kyoung An Park, Seon Bin Song, and You Kyoung Chung for insightful feedback on our work. This work was partly supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education, Science and Technology (NRF-2022M3H3A106307411, NRF-2023M3K5A1094805, NRF-2023M3K5A1094813, NRF-2020M3H3A1110365). This work was also partly supported by the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korean government (MSIP) (No. 2019-0-00003, Research and Development of Core Technologies for Programming, Running, Implementing and Validating of Fault-Tolerant Quantum Computing System). The Ministry of Trade, Industry, and Energy (MOTIE), Korea, also partly supported this research under the Industrial Innovation Infrastructure Development Project (Project No. RS-2024-00466693).

Appendix A: MAFFT Commands

In MAFFT, the `--distout` option enables obtaining pairwise distances between each pair of input sequences [6]. The following command was used to extract initial distances from MAFFT L-INS-i:

```
mafft --localpair --maxiterate 0 \
--distout input_file
```

Although the `--distout` option is present in the source code, it is a hidden feature that is not mentioned in official manuals or papers. Only a few studies have referenced its use [6]. This command generates files with the `.hat2` extension that contains distance information in the input file directory. Note that the command does not specify a directory for the output file. By default, the code prints distances to three decimal places. To print more decimal places, the `WriteFloatHat2_pointer_halfmtx` function in the file `io.c` has to be manually modified. This modification is limited to adjusting the number of decimal places printed and does not affect the performance of the program.

MAFFT (ver. 7.526) L-INS-i was used for MSAs in Section IV. However, our guide tree construction method is applicable to other MSA tools. We used the following command for L-INS-i [22] in Section IV:

```
mafft --localpair --maxiterate 2 \
    --treein input_guide_tree_file \
    input_file > output_file
```

Appendix B: Parallelizing Distance Matrix Computation

A simple method for parallel computation of distance matrices is suggested here. First, a given set of sequences is sorted in increasing order of length because distance calculations for several long sequences can be highly expensive compared with those for numerous short sequences. Calculating the distance between two sequences of lengths m and n requires $O(m+n)$ and $O(mn)$ time complexities for approximate measures [21, 91] and exact measures [30], respectively. As each sequence length is a positive integer with at most k digits, we can sort them in $O(kN) \approx O(N)$ time complexity using radix sort.

Assume that the sorted sequences are labeled as $s_1, s_2, \ldots, s_N$ for N input sequences, where s_1 is the shortest among them. Suppose that the number of threads available is T, which is smaller or equal to $N-1$. We define the following two values q and r:

$$q = \left\lfloor \frac{N-1}{T} \right\rfloor, \tag{B1}$$

$$r = (N-1) \bmod T. \tag{B2}$$

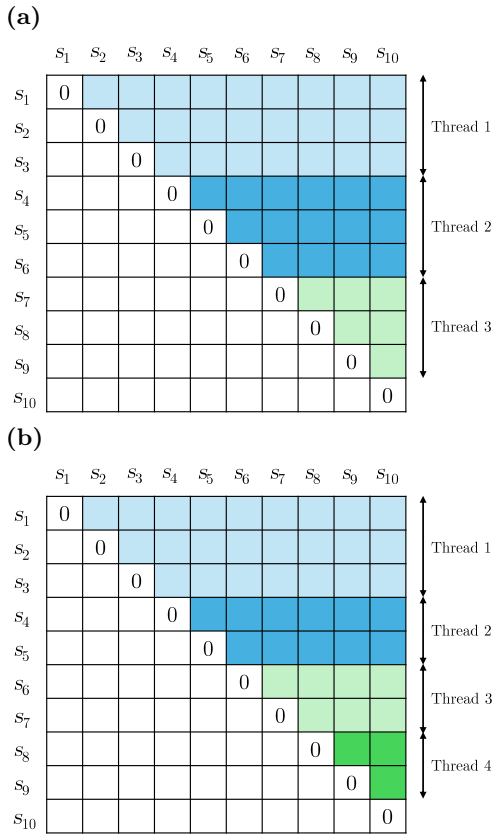

FIG. 8. Examples of parallel computation of distance matrices. For a given set of 10 sequences, (a) uses three threads, and (b) uses four threads.

We also define l_i and J_i, which are the labels of the first row assigned to the i-th thread and the set of distance computation jobs assigned to the i-thread, respectively.

There are two cases: (1) $r = 0$, and (2) $r > 0$. When $r = 0$, each thread is equally assigned q number of rows:

$$l_i = q(i-1) + 1, \tag{B3}$$

$$J_i = \bigcup_{j=q(i-1)+1}^{qi} \{(s_j, s_k) : j < k \leq N\}. \tag{B4}$$

When $r > 0$, the first r threads are assigned $q+1$ number of rows and the other threads have q number of rows:

$$l_i = \begin{cases} (q+1)(i-1)+1 & (1 \leq i \leq r) \\ q(i-1)+r+1 & (r < i \leq T), \end{cases} \tag{B5}$$

$$J_i = \begin{cases} \bigcup_{j=(q+1)(i-1)+1}^{(q+1)(i-1)+q+1} \{(s_j, s_k) : j < k \leq N\} & (1 \leq i \leq r) \\ \bigcup_{j=q(i-1)+r+1}^{q(i-1)+r+q} \{(s_j, s_k) : j < k \leq N\} & (r < i \leq T). \end{cases} \tag{B6}$$

Consider the examples shown in Fig. 8, where the number of sequences is 10. The number of threads is 3 in Fig. 8(a), so $q = 3$ and $r = 0$. As $r = 0$, each thread is assigned an equal number of rows ($q = 3$) to compute. Applying Eq. B3, we obtain

$$\begin{aligned} l_1 &= 3(1-1)+1 = 1, \\ l_2 &= 3(2-1)+1 = 4, \\ l_3 &= 3(3-1)+1 = 7. \end{aligned}$$

Thus, the second thread begins calculations from s_4 (fourth row) to s_6 (sixth row). Using Eq. B4, the set of jobs for the second thread is determined as follows:

$$\begin{aligned} J_2 = &\bigcup_{j=3(2-1)+1}^{6} \{(s_j, s_k) : j < k \leq N\} \\ =&\{(s_4, s_k) : 4 < k \leq N\} \cup \{(s_5, s_k) : 5 < k \leq N\} \\ &\cup \{(s_6, s_k) : 6 < k \leq N\} \\ =&\{(s_4, s_5), (s_4, s_6), (s_4, s_7), (s_4, s_8), (s_4, s_9), (s_4, s_{10}), \\ &(s_5, s_6), (s_5, s_7), (s_5, s_8), (s_5, s_9), (s_5, s_{10}), \\ &(s_6, s_7), (s_6, s_8), (s_6, s_9), (s_6, s_{10})\}. \end{aligned}$$

As shown in Fig. 8(a), thread 1 processes the first three rows, which involves a lighter workload compared with threads 2 and 3. Thus, the number of distance calculations assigned to thread 1 is the largest, whereas thread 3 has the smallest workload.

Fig. 8(b) shows where $q = 2$ and $r = 1$. If each thread is assigned two rows, one row will remain unassigned. Therefore, an additional row is assigned to thread 1. By explicitly substituting $q = 2$ and $r = 1$, one can verify that Eq. B5 and Eq. B6 are correct.

This parallel computing scheme has the same time complexity as non-parallel computing, scaling quadratically with respect to the problem size. However, if $N-1$ threads are used, this scheme can effectively reduce the computational time by a factor of $O(N)$.

Appendix C: D-Wave Implementation

A hybrid solver (`hybrid_binary_quadratic_model_version2p`) was used to solve the TSP. The embedding of binary variables to physical qubits is automatically handled by the hybrid solver; therefore, we did not customize or optimize the embedding.

Appendix D: BAliBASE Core Block Data

BAliBASE provides MSA reference results indicating core blocks in XML format files [96]. The core block data are located at the end of these files within the `<colsco-data>` tag. A value of 1 indicates a column that is determined to be a core block.

[1] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, *et al.*, Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega, Mol. Syst. Biol. **7**, 539 (2011).

[2] R. C. Edgar and S. Batzoglou, Multiple sequence alignment, Curr. Opin. Struct. Biol. **16**, 368 (2006).

[3] B. G. Hall, Building phylogenetic trees from molecular data with mega, Mol. Biol. Evol. **30**, 1229 (2013).

[4] S. Kumar, G. Stecher, M. Li, C. Knyaz, and K. Tamura, Mega x: molecular evolutionary genetics analysis across computing platforms, Mol. Biol. Evol. **35**, 1547 (2018).

[5] D. N. Gilbert *et al.*, *The Sanford guide to antimicrobial therapy* (Antimicrobial Therapy, Inc., 2023).

[6] K. Takeda, S. Sakakibara, K. Yamashita, D. Motooka, S. Nakamura, M. A. El Hussien, J. Katayama, Y. Maeda, M. Nakata, S. Hamada, *et al.*, Allergic conversion of protective mucosal immunity against nasal bacteria in patients with chronic rhinosinusitis with nasal polyposis, J. Allergy Clin. Immunol. **143**, 1163 (2019).

[7] C. L. Pierri, G. Parisi, and V. Porcelli, Computational approaches for protein function prediction: a combined strategy from multiple sequence alignment to molecular docking-based virtual screening, BBA-Proteins Proteomics **1804**, 1695 (2010).

[8] D. P. Ryan and J. M. Matthews, Protein–protein interactions in human disease, Curr. Opin. Struct. Biol. **15**, 441 (2005).

[9] D. Collias and C. L. Beisel, Crispr technologies and the search for the pam-free nuclease, Nat. Commun. **12**, 555 (2021).

[10] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. R. Nelson, A. Bridgland, *et al.*, Improved protein structure prediction using potentials from deep learning, Nature **577**, 706 (2020).

[11] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, Highly accurate protein structure prediction with alphafold, Nature **596**, 583 (2021).

[12] M. Ekeberg, C. Lövkvist, Y. Lan, M. Weigt, and E. Aurell, Improved contact prediction in proteins: using pseudolikelihoods to infer potts models, Phys. Rev. E **87**, 012707 (2013).

[13] W. Just, Computational complexity of multiple sequence alignment with sp-score, J. Comput. Biol. **8**, 615 (2001).

[14] L. Wang and T. Jiang, On the complexity of multiple sequence alignment, J. Comput. Biol. **1**, 337 (1994).

[15] A. Löytynoja, A. J. Vilella, and N. Goldman, Accurate extension of multiple sequence alignments using a phylogeny-aware graph algorithm, Bioinformatics **28**, 1684 (2012).

[16] A. Löytynoja and N. Goldman, An algorithm for progressive multiple alignment of sequences with insertions, Proc. Natl. Acad. Sci. U.S.A. **102**, 10557 (2005).

[17] A. Loytynoja and N. Goldman, Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis, Science **320**, 1632 (2008).

[18] M. A. Larkin, G. Blackshields, N. P. Brown, R. Chenna, P. A. McGettigan, H. McWilliam, F. Valentin, I. M. Wallace, A. Wilm, R. Lopez, *et al.*, Clustal w and clustal x version 2.0, Bioinformatics **23**, 2947 (2007).

[19] C. Grasso and C. Lee, Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems, Bioinformatics **20**, 1546 (2004).

[20] B. Morgenstern, K. Frech, A. Dress, and T. Werner, Dialign: finding local similarities by multiple sequence alignment, Bioinformatics **14**, 290 (1998).

[21] T. Lassmann and E. L. Sonnhammer, Kalign–an accurate and fast multiple sequence alignment algorithm, BMC Bioinformatics **6**, 298 (2005).

[22] K. Katoh and D. M. Standley, Mafft multiple sequence alignment software version 7: improvements in performance and usability, Mol. Biol. Evol. **30**, 772 (2013).

[23] Y. Liu, B. Schmidt, and D. L. Maskell, Msaprobs: multiple sequence alignment based on pair hidden markov models and partition function posterior probabilities, Bioinformatics **26**, 1958 (2010).

[24] R. C. Edgar, Muscle: multiple sequence alignment with high accuracy and high throughput, Nucleic Acids Res. **32**, 1792 (2004).

[25] R. C. Edgar, Muscle: a multiple sequence alignment method with reduced time and space complexity, BMC Bioinformatics **5**, 113 (2004).

[26] U. Roshan and D. R. Livesay, Probalign: multiple sequence alignment using partition function posterior probabilities, Bioinformatics **22**, 2715 (2006).

[27] C. B. Do, M. S. P. Mahabhashyam, M. Brudno, and S. Batzoglou, Probcons: Probabilistic consistency-based multiple sequence alignment, Genome Res. **15**, 330 (2005).

[28] C. Notredame, D. G. Higgins, and J. Heringa, T-coffee: A novel method for fast and accurate multiple sequence alignment, J. Mol. Biol. **302**, 205 (2000).

[29] D.-F. Feng and R. F. Doolittle, Progressive sequence alignment as a prerequisitetto correct phylogenetic trees, J. Mol. Evol. **25**, 351 (1987).

[30] S. R. Eddy, What is dynamic programming?, Nat. Biotechnol. **22**, 909 (2004).

[31] W.-S. Juang and S.-F. Su, Multiple sequence alignment using modified dynamic programming and particle swarm optimization, J. Chin. Inst. Eng. **31**, 659 (2008).

[32] S. B. Needleman and C. D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, J. Mol. Biol. **48**, 443 (1970).

[33] T. F. Smith, M. S. Waterman, *et al.*, Identification of common molecular subsequences, J. Mol. Biol. **147**, 195 (1981).

[34] Q. Zhan, Y. Ye, T.-W. Lam, S.-M. Yiu, Y. Wang, and H.-F. Ting, Improving multiple sequence alignment by using better guide trees, in *BMC Bioinformatics*, Vol. 16 (2015).

[35] M. Chatzou, C. Magis, J.-M. Chang, C. Kemena, G. Bussotti, I. Erb, and C. Notredame, Multiple sequence alignment modeling: methods and applications, Brief. Bioinform. **17**, 1009 (2016).

[36] A. Stamatakis, Raxml-vi-hpc: maximum likelihood-based phylogenetic analyses with thousands of taxa and

mixed models, Bioinformatics **22**, 2688 (2006).
[37] Z. Yang, Phylogenetic analysis using parsimony and likelihood methods, J. Mol. Evol. **42**, 294 (1996).
[38] J. P. Huelsenbeck, B. Larget, R. E. Miller, and F. Ronquist, Potential applications and pitfalls of bayesian inference of phylogeny, Syst. Biol. **51**, 673 (2002).
[39] Y. Loewenstein, E. Portugaly, M. Fromer, and M. Linial, Efficient algorithms for accurate hierarchical clustering of huge datasets: tackling the entire protein space, Bioinformatics **24**, i41 (2008).
[40] A. Krause, J. Stoye, and M. Vingron, Large scale hierarchical clustering of protein sequences, BMC Bioinformatics **6**, 15 (2005).
[41] R. Agarwala, V. Bafna, M. Farach, M. Paterson, and M. Thorup, On the approximability of numerical taxonomy (fitting distances by tree metrics), SIAM J. Comput. **28**, 1073 (1998).
[42] D. Catanzaro, M. Labbé, R. Pesenti, and J.-J. Salazar-González, Mathematical models to reconstruct phylogenetic trees under the minimum evolution criterion, Networks **53**, 126 (2009).
[43] W. H. E. Day, Computational complexity of inferring phylogenies from dissimilarity matrices, Bull. Math. Biol. **49**, 461 (1987).
[44] D. Catanzaro, The minimum evolution problem: Overview and classification, Networks **53**, 112 (2009).
[45] N. Saitou and M. Nei, The neighbor-joining method: a new method for reconstructing phylogenetic trees, Mol. Biol. Evol. **4**, 406 (1987).
[46] R. R. Sokal and C. D. Michener, A statistical method for evaluating systematic relationships, Univ. Kansas Sci. Bull. **38**, 1409 (1958).
[47] K. Howe, A. Bateman, and R. Durbin, Quicktree: building huge neighbour-joining trees of protein sequences, Bioinformatics **18**, 1546 (2002).
[48] J. Evans, L. Sheneman, and J. Foster, Relaxed neighbor joining: a fast distance-based phylogenetic tree construction method, J. Mol. Evol. **62**, 785 (2006).
[49] M. Simonsen, T. Mailund, and C. N. S. Pedersen, Rapid neighbour-joining, in *Algorithms in Bioinformatics: 8th International Workshop, WABI 2008, Karlsruhe, Germany, September 15-19, 2008, Proceedings* (Springer Berlin Heidelberg, 2008) pp. 113–122.
[50] I. Elias and J. Lagergren, Fast neighbor joining, Theor. Comput. Sci. **410**, 1993 (2009).
[51] K. Katoh and H. Toh, Parttree: an algorithm to build an approximate tree from a large number of unaligned sequences, Bioinformatics **23**, 372 (2007).
[52] G. Blackshields, F. Sievers, W. Shi, A. Wilm, and D. G. Higgins, Sequence embedding for fast construction of guide trees for multiple sequence alignment, Algorithms. Mol. Biol. **5**, 21 (2010).
[53] A. B. Finnila, M. A. Gomez, C. Sebenik, C. Stenson, and J. D. Doll, Quantum annealing: A new method for minimizing multidimensional functions, Chem. Phys. Lett. **219**, 343 (1994).
[54] A. Rajak, S. Suzuki, A. Dutta, and B. K. Chakrabarti, Quantum annealing: An overview, Phil. Trans. R. Soc. A. **381**, 20210417 (2023).
[55] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem, Science **292**, 472 (2001).
[56] J. Brooke, D. Bitko, T. F. Rosenbaum, and G. Aeppli, Quantum annealing of a disordered magnet, Science **284**, 779 (1999).
[57] S. Mukherjee and B. K. Chakrabarti, Multivariable optimization: Quantum annealing and computation, Eur. Phys. J. Spec. Top. **224**, 17 (2015).
[58] E. Crosson and A. W. Harrow, Simulated quantum annealing can be exponentially faster than classical simulated annealing, in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE, 2016) pp. 714–723.
[59] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, *et al.*, Quantum annealing with manufactured spins, Nature **473**, 194 (2011).
[60] R. Harris, M. W. Johnson, T. Lanting, A. J. Berkley, J. Johansson, P. Bunyk, E. Tolkacheva, E. Ladizinsky, N. Ladizinsky, T. Oh, *et al.*, Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor, Phys. Rev. B **82**, 024511 (2010).
[61] S. Ebadi, T. T. Wang, H. Levine, A. Keesling, G. Semeghini, A. Omran, D. Bluvstein, R. Samajdar, H. Pichler, W. W. Ho, *et al.*, Quantum phases of matter on a 256-atom programmable quantum simulator, Nature **595**, 227 (2021).
[62] P. Scholl, M. Schuler, H. J. Williams, A. A. Eberharter, D. Barredo, K.-N. Schymik, V. Lienhard, L.-P. Henry, T. C. Lang, T. Lahaye, *et al.*, Quantum simulation of 2d antiferromagnets with hundreds of rydberg atoms, Nature **595**, 233 (2021).
[63] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, Quantum annealing for industry applications: Introduction and review, Rep. Prog. Phys. **85**, 104001 (2022).
[64] M. Kim, K. Kim, J. Hwang, E.-G. Moon, and J. Ahn, Rydberg quantum wires for maximum independent set problems, Nat. Phys. **18**, 755 (2022).
[65] M. Steffen, W. van Dam, T. Hogg, G. Breyta, and I. Chuang, Experimental implementation of an adiabatic quantum optimization algorithm, Phys. Rev. Lett. **90**, 067903 (2003).
[66] W. Lechner, P. Hauke, and P. Zoller, A quantum annealing architecture with all-to-all connectivity from local interactions, Sci. Adv. **1**, e1500838 (2015).
[67] T. Albash and D. A. Lidar, Demonstration of a scaling advantage for a quantum annealer over simulated annealing, Phys. Rev. X **8**, 031016 (2018).
[68] A. D. King, J. Raymond, T. Lanting, S. V. Isakov, M. Mohseni, G. Poulin-Lamarre, S. Ejtemaee, W. Bernoudy, I. Ozfidan, A. Y. Smirnov, *et al.*, Scaling advantage over path-integral monte carlo in quantum simulation of geometrically frustrated magnets, Nat. Commun. **12**, 1113 (2021).
[69] A. Rzhetsky and M. Nei, A simple method for estimating and testing minimum-evolution trees, Mol. Biol. Evol. **9**, 945 (1992).
[70] A. Rzhetsky and M. Nei, Theoretical foundation of the minimum-evolution method of phylogenetic inference, Mol. Biol. Evol. **10**, 1073 (1993).
[71] O. Gascuel and M. Steel, Neighbor-joining revealed, Mol. Biol. Evol. **23**, 1997 (2006).
[72] L. L. Cavalli-Sforza and A. W. Edwards, Phylogenetic analysis. models and estimation procedures, Am. J. Hum. Genet. **19**, 233 (1967).

[73] A. Drummond and A. G. Rodrigo, Reconstructing genealogies of serial samples under the assumption of a molecular clock using serial-sample upgma, Mol. Biol. Evol. **17**, 1807 (2000).

[74] S. Kumar, Molecular clocks: four decades of evolution, Nat. Rev. Genet. **6**, 654 (2005).

[75] E. Zuckerkandl and L. Pauling, *Molecular Disease, Evolution, and Genic Heterogeneity* (Academic Press, 1962).

[76] E. Zuckerkandl and L. Pauling, Evolutionary divergence and convergence in proteins, in *Evolving Genes and Proteins* (Academic Press, 1965) pp. 97–166.

[77] C. Korostensky and G. H. Gonnet, Using traveling salesman problem algorithms for evolutionary tree construction, Bioinformatics **16**, 619 (2000).

[78] A. Ambainis, K. Balodis, J. Iraids, M. Kokainis, K. Prūsis, and J. Vihrovs, Quantum speedups for exponential-time dynamic programming algorithms, in *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (SIAM, 2019) pp. 1783–1793.

[79] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm (2014), arXiv:1411.4028 [quant-ph].

[80] J. Choi and J. Kim, A tutorial on quantum approximate optimization algorithm (qaoa): Fundamentals and applications, in *2019 international conference on information and communication technology convergence (ICTC)* (IEEE, 2019) pp. 138–142.

[81] S. Y. Madsen, F. K. Marqversen, S. E. Rasmussen, and N. T. Zinner, Multi-sequence alignment using the quantum approximate optimization algorithm (2023), arXiv:2308.12103 [quant-ph].

[82] O. B. Lindvall, *Quantum Methods for Sequence Alignment and Metagenomics*, Ph.D. thesis, KTH Royal Institute of Technology (2019).

[83] M. Lee, Modified multiple sequence alignment algorithm on quantum annealers (maq) (2024), arXiv:2403.17979 [quant-ph].

[84] C. Korostensky and G. Gonnet, Near optimal multiple sequence alignments using a traveling salesman problem approach, in *6th International Symposium on String Processing and Information Retrieval. 5th International Workshop on Groupware (Cat. No. PR00268)* (IEEE, 1999) pp. 105–114.

[85] K. Giannakis, C. Papalitsas, G. Theocharopoulou, S. Fanarioti, and T. Andronikos, A quantum-inspired optimization heuristic for the multiple sequence alignment problem in bio-computing, in *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)* (IEEE, 2019) pp. 1–8.

[86] K. Katoh and D. M. Standley, A simple method to control over-alignment in the mafft multiple sequence alignment program, Bioinformatics **32**, 1933 (2016).

[87] S. Yamada, O. Gotoh, and H. Yamana, Improvement in accuracy of multiple sequence alignment using novel group-to-group sequence alignment algorithm with piecewise linear gap cost, BMC Bioinformatics **7**, 524 (2006).

[88] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt, A model of evolutionary change in proteins, Atlas Protein Seq. Struct. **5**, 345 (1978).

[89] J. D. Thompson, D. G. Higgins, and T. J. Gibson, Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, Nucleic Acids Res. **22**, 4673 (1994).

[90] K. Katoh, K. Misawa, K.-i. Kuma, and T. Miyata, Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform, Nucleic Acids Res. **30**, 3059 (2002).

[91] S. Wu and U. Manber, Fast text searching: allowing errors, Commun. ACM **35**, 83 (1992).

[92] B. Chowdhury and G. Garai, A review on multiple sequence alignment from the perspective of genetic algorithm, Genomics **109**, 419 (2017).

[93] F. Sievers, G. M. Hughes, and D. G. Higgins, Systematic exploration of guide-tree topology effects for small protein alignments, BMC Bioinformatics **15**, 338 (2014).

[94] K. Boyce, F. Sievers, and D. G. Higgins, Simple chained guide trees give high-quality protein multiple sequence alignments, Proc. Natl. Acad. Sci. U.S.A. **111**, 10556 (2014).

[95] J. Felsenstein, The number of evolutionary trees, Syst. Biol. **27**, 27 (1978).

[96] J. D. Thompson, P. Koehl, R. Ripp, and O. Poch, Balibase 3.0: latest developments of the multiple sequence alignment benchmark, Proteins **61**, 127 (2005).

[97] A. Lucas, Ising formulations of many np problems, Front. Physics **2**, 5 (2014).

[98] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms* (MIT press, 2022).

[99] I. Gronau and S. Moran, Optimal implementations of upgma and other common clustering algorithms, Inf. Process. Lett. **104**, 205 (2007).

[100] G. W. Milligan, Ultrametric hierarchical clustering algorithms, Psychometrika **44**, 343 (1979).

[101] G. N. Lance and W. T. Williams, A general theory of classificatory sorting strategies: 1. hierarchical systems, Comput. J. **9**, 373 (1967).

[102] F. Sievers and D. G. Higgins, Clustal omega for making accurate alignments of many protein sequences, Protein Sci. **27**, 135 (2018).

[103] K. Katoh, J. Rozewicki, and K. D. Yamada, Mafft online service: multiple sequence alignment, interactive sequence choice and visualization, Brief. Bioinform. **20**, 1160 (2019).

[104] J. D. Thompson, F. Plewniak, and O. Poch, A comprehensive comparison of multiple sequence alignment programs, Nucleic Acids Res. **27**, 2682 (1999).

[105] A. Virkamäki, K. Ueki, and C. R. Kahn, Protein–protein interaction in insulin signaling and the molecular mechanisms of insulin resistance, J. Clin. Invest. **103**, 931 (1999).

[106] L. Chen and A. Kashina, Post-translational modifications of the protein termini, Front. Cell. Dev. Biol. **9**, 719590 (2021).

[107] T. V. Kalvik and T. Arnesen, Protein n-terminal acetyltransferases in cancer, Oncogene **32**, 269 (2013).

[108] V. N. Uversky, The most important thing is the tail: multitudinous functionalities of intrinsically disordered protein termini, FEBS Lett. **587**, 1891 (2013).

[109] B. Mészáros, I. Simon, and Z. Dosztányi, The expanding view of protein–protein interactions: complexes involving intrinsically disordered proteins, Phys. Biol. **8**, 035003 (2011).

[110] R. Dawson, L. Müller, A. Dehner, C. Klein, H. Kessler, and J. Buchner, The n-terminal domain of p53 is na-

tively unfolded, J. Mol. Biol. **332**, 1131 (2003).

[111] C. J. Oldfield, J. Meng, J. Y. Yang, M. Q. Yang, V. N. Uversky, and A. K. Dunker, Flexible nets: disorder and induced fit in the associations of p53 and 14-3-3 with their partners, BMC Genomics **9**, S1 (2008).

[112] K. Katoh and H. Toh, Recent developments in the mafft multiple sequence alignment program, Brief. Bioinform. **9**, 286 (2008).

[113] N. Nikolova and J. Jaworska, Approaches to measure chemical similarity–a review, QSAR Comb. Sci. **22**, 1006 (2003).

[114] R. D. Brown and Y. C. Martin, Use of structure-activity data to compare structure-based clustering methods and descriptors for use in compound selection, J. Chem. Inf. Comput. Sci. **36**, 572 (1996) .