The Journal of Supercomputing (2026) 82:324
https://doi.org/10.1007/s11227-026-08465-x

RESEARCH

Reconstruction of phylogenetic trees via graph-splitting using quantum computing

Nicolás Fernández-Otero[1] · Tomás F. Pena[1,2] · Juan C. Pichel[1,2]

Received: 15 October 2025 / Accepted: 18 March 2026

Abstract
Quantum computing applies principles of quantum mechanics, such as superposition and entanglement, to process information with exponential parallelism. This paradigm offers significant computational advantages over classical methods, particularly for NP-hard problems like phylogenetic tree reconstruction in evolutionary biology. Phylogenetic trees model the evolutionary relationships among species or genes, and their reconstruction is computationally challenging as the number of possible topologies grows exponentially with the number of taxa. To address this, biologists often rely on heuristic methods; however, recent work has shown that recursive graph-cut techniques can achieve high accuracy in phylogenetic inference, though at high computational cost. In this study, we present a quantum algorithm based on the normalized cut (N_{cut}) criterion, enabling efficient recursive graph partitioning. Implemented using Quantum Annealing (QA) and the Quantum Approximate Optimization Algorithm (QAOA), demonstrating promising results on real quantum hardware for complex bioinformatics tasks.

Keywords Phylogenetic tree · Quantum annealing · Quantum approximate optimization algorithm · Min_{cut} · N_{cut}

Tomás F. Pena and Juan C. Pichel have contributed equally to this work.

✉ Nicolás Fernández-Otero
nicolas.fernandez.otero@rai.usc.es

Tomás F. Pena
tf.pena@usc.es

Juan C. Pichel
juancarlos.pichel@usc.es

[1] Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, Santiago de Compostela, Spain

[2] Departamento de Electrónica e Computación, Universidade de Santiago de Compostela, Santiago de Compostela, Spain

Published online: 06 April 2026

1 Introduction

Phylogenetic trees, also known as phylogenies or evolutionary trees, are structures that represent evolutionary relationships among a set of organisms based on anatomical, physiological, and molecular characteristics. Each branch point, or node, in the tree denotes an individual, while the branching patterns illustrate how lineages diverge over time. More specifically, the terminal (leaf) nodes correspond to individual taxa or samples, whereas the internal nodes represent their common ancestors. These types of trees are the basic structures necessary to compute differences between species and to analyze those differences statistically [1], playing a crucial role in fields such as genomics. Phylogenetic trees can be reconstructed using various types of data, including genetic sequences, which will be the focus of this study. Genetic sequences are strings that represent DNA, RNA, or protein sequences. For each type, a simplified alphabet is used to represent different nucleotides (for DNA and RNA) or amino acids (for proteins). For example, nucleotide sequences are represented using the alphabet $\mathcal{A} = G, A, C, T$, and a possible sequence could be $ATTACAGGA$. Reconstruction of phylogenetic trees is a computationally challenging task, for which exact solutions are often infeasible. To address this problem, approximate methods are commonly employed. Among these approaches, algorithms such as neighbor-joining [2], maximum parsimony [3], and UPGMA [4] have been widely adopted due to their efficiency and effectiveness. However, in this work, we focus on an alternative paradigm to these bottom-up reconstruction algorithms, namely graph-splitting methods [5], which offer a different perspective on phylogenetic reconstruction.

Graph-splitting methods rely on a top-down approach that recursively partitions a distance matrix (a matrix representing the genetic distances between individuals). This recursive decomposition provides a systematic way to simplify the problem into smaller subproblems, making it well-suited for phylogenetic inference. In this context, one possible implementation of such an approach can be found in [6], where the authors develop a graph-splitting method based on stochastic searches (the Markov chain Monte Carlo method) and use the normalized cut (N_{cut}) to obtain a near-optimal cut. This method, named the "Normalized-Minimum cut by Digital Annealer (NMcutDA)," was run on the Fujitsu Digital Annealer (DA), which is a quantum-inspired solver designed to solve QUBO problems [7]. However, the DA, while efficient, cannot compare to the capabilities of real quantum computers. In this work, we implement the NMcutDA method using real quantum computing techniques such as Quantum Annealing (QA) [8] and the Quantum Approximate Optimization Algorithm (QAOA) [9]. These methods, particularly QA, may provide precise solutions while potentially requiring significantly less time than DA. Our quantum approach, developed to reconstruct phylogenetic trees and referred to as QNMcut, solves an optimization problem represented in a QUBO formulation, which can be transformed into an Ising model and thus solved using quantum optimization methods. The method can be directly executed on real quantum hardware, and the solutions obtained by QNMcut will be compared with those obtained from other reconstruction methods.

The remainder of this paper is organized as follows. Section 2 provides the necessary background in phylogenetics and quantum computing. Section 3 introduces the proposed QNMcut method for reconstructing phylogenetic trees using quantum opti-

mization techniques. Section 5 presents the experimental setup and results. Finally, Sect. 7 discusses the conclusions and the future work drawn from this work.

2 Background and related work

This section provides an introduction to the background of the paper, focusing on the reconstruction of phylogenetic trees and the quantum optimization methods employed in this study: Quantum Annealing (QA) and the Quantum Approximate Optimization Algorithm (QAOA).

2.1 Phylogenetic tree reconstruction

To reconstruct phylogenetic trees, both distance matrix-based methods and character-based methods can be defined. Character-based methods rely on specific anatomical, physiological, or molecular traits to infer evolutionary relationships and include algorithms such as maximum parsimony and maximum likelihood. The maximum parsimony method selects the tree topology that minimizes the total number of evolutionary changes required to explain the observed character states across taxa. In contrast, the maximum likelihood method identifies the tree that maximizes the probability of observing the given data under a specified model of evolution. Both methods are capable of yielding precise solutions; however, in this work, we will focus on the distance matrix-based approaches.

Distance matrix methods utilize a matrix that represents the degree of biological distance between sequences. Using this matrix, multiple algorithms can be defined. Due to the complexity of the problem, all currently used reconstruction algorithms are approximate methods that employ various heuristics to generate phylogenetic trees. Methods such as neighbor-joining [2], UPGMA [4], and graph splitting [5] are heuristic approaches that can efficiently and accurately generate phylogenies using a distance matrix. In particular, the neighbor-joining and UPGMA methods reconstruct trees using a bottom-up approach by continuously clustering closely related nodes. In UPGMA, clustering is performed directly using the distance matrix, assuming a constant rate of evolution. In contrast, the neighbor-joining method constructs a modified matrix that accounts for varying rates of evolution, with the smallest element identifying the pair of nodes to be joined at each step. This allows it to produce more accurate trees under conditions of variable evolutionary rates. These methods, along with maximum likelihood, are widely regarded as the de facto standards for phylogenetic tree reconstruction. Several state-of-the-art software tools implement these algorithms: UPGMA is employed by Clustal Ω [10], neighbor-joining is used by FastTree 2 [11] to reconstruct an initial tree, and maximum likelihood is employed by RAxML [12], VeryFastTree [13], and also by FastTree 2.

In contrast, graph-splitting methods use a top-down approach that works by recursively splitting the elements into two groups. The splitting method aims to maximize the distance between groups or minimize the distance between sequences within the same group. Solving this optimization problem involves evaluating a vast number of

possible cutting configurations, each corresponding to a different partitioning of the tree's structure. As the number of potential cut points increases, the solution space grows exponentially, leading to a combinatorial explosion that poses significant computational challenges. Consequently, approximate solutions must be used. One of the most common approximate solutions is spectral clustering, a method that transforms the cutting problem into a generalized eigenvalue problem [14]. An example of the use of spectral clustering for phylogeny reconstruction can be found in [15]. In other related work [6], the authors develop a graph-splitting method based on stochastic searches (the Markov chain Monte Carlo method) that uses the normalized cut (N_{cut}) criterion [14] to obtain a near-optimal partition. The normalized cut approach addresses limitations associated with minimum cuts, which only minimize the similarity between the resulting subgraphs. Rather than considering the total edge weight connecting the two partitions, the normalized cut computes the cost of a cut as a fraction of the total edge connections to all nodes in the graph, providing a more balanced partitioning. As noted in the introduction, Onodera et al. work [6] served as the foundation for the present study. To address the N_{cut} problem, the original paper uses a digital annealer, a quantum-inspired solver designed to tackle QUBO problems. However, the results obtained using the digital annealer could potentially be improved by employing real quantum methods for the optimization tasks, which is the focus of our work. In the following section, we provide an overview of two such methods: Quantum Annealing and the Quantum Approximate Optimization Algorithm (QAOA).

2.2 QA and QAOA

In the context of quantum computing, two prominent methods are employed to tackle optimization problems: Quantum Annealing and the Quantum Approximate Optimization Algorithm. Both approaches aim to find the ground state of a *cost Hamiltonian* that encodes the optimal solution.

QA operates by evolving a quantum system according to the time-dependent Schrödinger equation. In this framework, the Hamiltonian represents the total energy of the system, and its ground state corresponds to the configuration with the lowest possible energy. The evolution starts from an easily prepared ground state of a *mixer Hamiltonian* (H_m), which introduces quantum fluctuations that allow the system to explore different configurations. The most common mixer Hamiltonian is defined as $H_m = \sum_{i=0}^{n-1} X_i$, where each X_i is a Pauli-X operator acting on qubit i. The ground state of this Hamiltonian is the uniform superposition $|\psi_0\rangle = |+\rangle^{\otimes n}$, representing an equal probability over all possible bit strings. This ground state can be prepared by applying Hadamard gates $H^{\otimes n}$ to the initial $|0\rangle^{\otimes n}$ state.

The evolution then slowly transitions to the cost Hamiltonian (H_c), which encodes the optimization problem to be solved, typically formulated in terms of an Ising model. The cost Hamiltonian defines the energy landscape of the problem, where the lowest-energy (ground) state corresponds to the optimal solution. The system evolves under a time-dependent Hamiltonian that interpolates between these two components:

$$H(t) = -A(t)H_m - B(t)H_c \tag{1}$$

where $A(t)$ and $B(t)$ are time-dependent scheduling functions that control the relative contributions of the mixer and cost Hamiltonians over time. Initially, $A(t)$ dominates and $B(t)$ is close to zero, keeping the system near the ground state of H_m. As the evolution proceeds, $A(t)$ decreases while $B(t)$ increases, gradually steering the system toward the ground state of H_c. If the process is sufficiently slow (adiabatic), the system remains close to its instantaneous ground state throughout the evolution, ideally reaching the optimal solution encoded in H_c at the end of the annealing schedule.

QAOA can be seen as a discretized version of QA. Instead of continuously evolving the system according to the Schrödinger equation, QAOA approximates this evolution by applying a finite sequence of quantum gates derived from the mixer Hamiltonian and the cost Hamiltonian. This sequence forms a parameterized quantum circuit, known as an *ansatz*, which alternates between unitary operations generated by H_m and H_c. Each layer of the circuit corresponds to a discrete time step in the annealing process, controlled by two sets of parameters, $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_{p-1})$ and $\boldsymbol{\gamma} = (\gamma_0, \ldots, \gamma_{p-1})$.

The final quantum state of the ansatz, denoted by $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$, is defined as:

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = \left(\prod_{n=0}^{p-1} e^{i\beta_n H_m} e^{i\gamma_n H_c} \right) |\psi_0\rangle, \tag{2}$$

where the initial state is, as in QA, the uniform superposition over all computational basis states.

To approximate the ground state of the system, QAOA employs a hybrid quantum–classical optimization procedure. After preparing the parameterized quantum state $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$, a classical optimizer iteratively adjusts the parameters $(\boldsymbol{\beta}, \boldsymbol{\gamma})$ to minimize the system's expected energy, computed as the expectation value of the cost Hamiltonian $\langle H_c \rangle = \langle \boldsymbol{\beta}, \boldsymbol{\gamma} | H_c | \boldsymbol{\beta}, \boldsymbol{\gamma} \rangle$. The optimal parameters correspond to the lowest measured energy, ideally approaching the ground state of H_c and thus the optimal solution to the underlying optimization problem.

Both methods, specially QA, can efficiently provide approximate solutions to optimization problems and are well-suited for addressing the optimization challenge posed by the graph-splitting procedure, as they are known to yield good results for matrix cut problems such as the maximum cut [16].

3 QNMcut method

In this section, we present the designed algorithm. The QNMcut algorithm is a recursive and iterative graph-splitting method that uses quantum algorithms to solve the optimization problem. As we will explain in detail below, the process begins with the definition of the similarity matrix. Then, we recursively solve the required Minimum Cut (Min_{cut}) problems and normalize them using the N_{cut} criterion until a tree is generated.

3.1 Alignment and similarity matrix generation

As outlined in the introduction, this study will utilize genetic sequences as the primary biological data source. To generate a similarity matrix from the genetic sequences of the individuals, we first need to perform a sequence alignment. A sequence alignment arranges the nucleotide or amino acid sequences of different organisms in a way that highlights regions of similarity, which may indicate functional, structural, or evolutionary relationships. By aligning the sequences, insertions, deletions, and substitutions can be identified, allowing the computation of pairwise distances that form the basis of the similarity matrix. Usually, multiple sequence alignment (MSA) is used. However, in this study, MSA was replaced by an all-to-all pairwise alignment (PSA). PSA offers more information than MSA and thus can be used to generate the similarity matrix [15].

The similarity matrix (or graph) is defined as a fully connected, all-to-all matrix that captures the relationships among all individuals within the population. To score the similarities between individuals, we will utilize the BLOSUM62 substitution matrix [17]. However, the score is not directly used as an element of the similarity matrix, but was normalized as described by [18]. The normalization score will be calculated using the following expression:

$$D_{ij} = \frac{\text{score}_{ij}}{[\text{score}_{ii} + \text{score}_{jj}]/2} * 100,$$

where score_{ij} represents the obtained similarity score of the individuals i and j from the population. This expression returns a value from 0 to 100, where a higher number represents higher similarity between sequences. To generate the similarity matrix D, we used the Biopython library [19], which provides the necessary functions to compute the similarity matrix from an alignment file.

3.2 Minimum cut of the similarity matrix

Viewing the similarity matrix from a graph-theoretical perspective, the graph-cut procedure can be interpreted as partitioning the graph into two subsets such that sequences within each subset exhibit high intra-group similarity. This corresponds to cutting the least similar edges between nodes, thereby framing the problem as an instance of the classical Minimum Cut (Min_{cut}) problem in graph theory.

As we will be using quantum optimization methods, it is necessary to define the Min_{cut} problem using a Quadratic Unconstrained Binary Optimization (QUBO) formulation so that it can be transformed into an equivalent Ising model representation. Therefore, the objective function of the Min_{cut} problem can be defined as follows for a population consisting of n individuals:

$$\text{Min}_{\text{cut}} = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} D_{ij}(x_i - x_j)^2, \quad x_i \in \{0, 1\}, \tag{3}$$

where x_i is the binary variable associated with sequence i. This value acts as a label indicating to which subgroup the sequence i belongs. The element D_{ij} represents the similarity between sequences i and j, as defined in the similarity matrix.

Using this formulation, if an edge between nodes i and j is cut (i.e., the two elements belong to different subgraphs), then $(x_i - x_j)^2 = 1$, and the corresponding similarity D_{ij} contributes to the total cut value. In the Min_{cut} problem, we seek to minimize the value of Eq. 3; therefore, the optimal solution minimizes the total weight of the edges that are cut.

However, this formulation poses two main issues. First, if no restrictions are added, the optimal solution is the trivial one, in which all elements are assigned to the same subset. In this case, no edges are cut and the objective function in Eq. 3 evaluates to zero, which represents the minimal possible value. This solution, although optimal in a mathematical sense, is meaningless for our purposes because it does not produce a valid partition of the graph. Second, it tends to generate *pectinate* (ladder-like) trees by repeatedly cutting small sets of nodes, resulting in highly unbalanced partitions that reduce the biological relevance of the reconstructed tree. To address these issues, an additional constraint is introduced in the formulation of Eq. 3, enforcing the solution to have exactly c elements, which results in the following expression:

$$\text{Min}_{\text{cut}}(c) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} D_{ij}(x_i - x_j)^2 + \alpha \left(\sum_{i=0}^{n-1} x_i - c \right)^2, \tag{4}$$

where α is a positive constant. The expression $\alpha \left(\sum_{i=0}^{n-1} x_i - c \right)^2$ acts as a penalty term that discourages solutions in which the number of variables labeled '1' differs from c. In our experiments, α was set to $\alpha = 100n$, where n is the size of the problem. This empirically chosen value ensures that the resulting output consistently attains size c, even when factoring quantum error. Larger values of α were not adopted in order to prevent excessive growth of the resulting quantities. Using this expression, the minimum cut for each possible size can be calculated, avoiding the issues mentioned above.

3.3 Normalization of the cut

To make use of the Min_{cut} formulation given in Eq. 4, the solutions obtained from the cuts have to be normalized, as the optimal solutions from each cut are in different scales. To do this, the normalized cut (N_{cut}) can be used [14]. The normalization of the cut is obtained by the following expression [6]:

$$N_{\text{cut}}(c) = \frac{\text{Min}_{\text{cut}}(c)}{\text{assoc}(A, V)} + \frac{\text{Min}_{\text{cut}}(c)}{\text{assoc}(B, V)}, \tag{5}$$

where $V = A \sqcup B$ is the graph obtained from the similarity matrix and A and B represent the subgraphs generated by the cut. The function $\text{assoc}(A, V)$ represents the similarity between the subgraph A and the original graph V and is calculated as

Algorithm 1 QNMcut.

1: Generation of the similarity matrix D from a PSA
2: **function** QNMcut(D)
3: **for** $i = 1, \ldots, \lfloor N/2 \rfloor$ **do**
4: Calculate $\text{Min}_{\text{cut}}(i) \leftarrow$ Using QA or QAOA
5: Calculate $N_{\text{cut}}(i)$
6: **end for**
7: Select A and B as the subgraphs with a lower $N_{\text{cut}}(c)$ value
8: QNMcut(A)
9: QNMcut(B)
10: **end function**

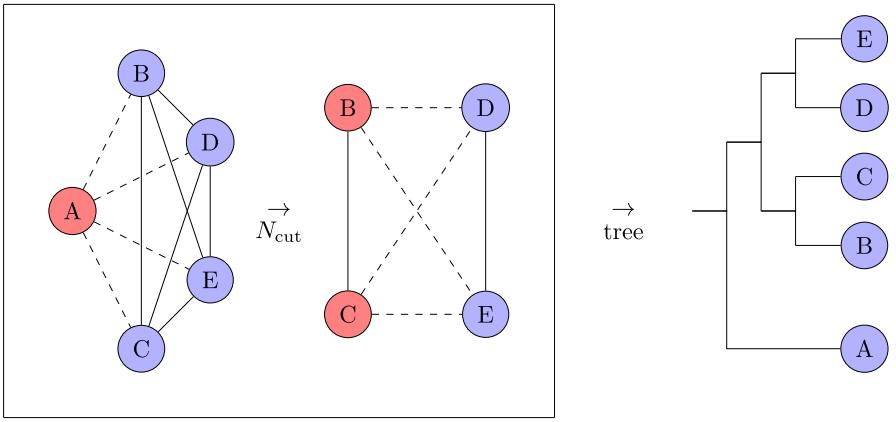

Fig. 1 Example of the QNMcut algorithm. Dashed lines indicate edges that have been cut, while different colors represent the assigned labels. In this example, the lowest N_{cut} value would have been obtained in the size-one cut. The procedure is then applied recursively to the resulting subgraphs. In this case, the second cut represents the final cut, as each selected subgraph has a size of two

$\text{assoc}(A, V) = \sum_{u \in A, t \in V} D_{ut}$. Therefore, the cuts are normalized by the sum of the rows of the original matrix corresponding to each of the subgraphs. This normalization enables a fair comparison of the energy associated with cuts of different sizes. By employing the normalized cut, more balanced phylogenetic trees are obtained compared to those generated using the minimum cut formulation in Eq. 3.

Now, we present the QNMcut algorithm for reconstructing a phylogenetic tree of size n. First, upon obtaining a similarity matrix, the value of $\text{Min}_{\text{cut}}(c)$ is calculated for all possible subgraph configurations ($c = 1, 2, \ldots, \lfloor \frac{n}{2} \rfloor$). With these values, the value of c that minimizes $N_{\text{cut}}(c)$ is selected. This process is then applied recursively to the resulting subgraphs until they are reduced to size 2.

In Algorithm 1, we define the basic structure of the method. Furthermore, Fig. 1 provides a visual example of how the algorithm works. In this example, only the cuts corresponding to the lowest N_{cut} values are shown. To identify these cuts, all possible tree configurations are evaluated. The $N_{\text{cut}}(c)$ is then calculated using Eq. 5, and the configuration with the lowest value is selected. This process is applied recursively until the resulting subgraphs have a size smaller than two. In this case, the selected

cut for the full graph is the size-one cut, and in the second step, the optimal size-two cut is chosen, thereby generating the tree.

In this approach, a quantum method is used to solve the Min_{cut} problem. QNMcut is the generic name of this method; however, to easily identify which quantum optimization technique is used, we will use the following notation:

- When referring to the QNMcut method that uses Quantum Annealing to solve the minimum cut problem, we denote it as **NMcutQA**. This variant benefits from the ability to be executed on specialized quantum annealing hardware, which is specifically optimized for large-scale combinatorial tasks. However, it is primarily limited by restricted qubit connectivity
- Analogously, when referring to the QNMcut method that uses QAOA to solve the minimum cut problem, we denote it as **NMcutQAOA**. This version offers the advantage of being compatible with general-purpose, gate-based quantum computers. Nevertheless, it faces challenges regarding high circuit depth and the overhead of classical parameter optimization

4 Experimental framework

This section presents the phylogenetic tree dataset, the evaluation metrics, and the quantum computing platforms used in the experiments.[1]

4.1 Sequence alignments and trees

Inferring the ground-truth phylogenetic tree is a challenging task, especially as the number of taxa grows. To address this, we used RAxML Grove [20], a curated database of phylogenetic trees generated with RAxML [12], which provides access to both the trees and the original sequence alignments used to reconstruct them. This resource enables the reliable retrieval of ground-truth trees along with their corresponding sequence data, thereby overcoming the common limitation of extracted trees lacking associated alignments. In total, 45 different alignments and their corresponding tree were considered.

Two main criteria were used in the selection process. First, the trees ranged from 8 to 29 taxa, with variable sequence lengths and including both nucleotide and protein sequences. This diversity was chosen because the goal was to verify whether the method performs well in general rather than to test its limitations. Second, the branch lengths, which represent evolutionary distances between taxa, were chosen to fall between 0.25 and 0.75, with a variance of less than 0.25. While these parameters allow for an initial verification of the method, the number of taxa is inherently constrained by the limitations of current quantum hardware. As scaling to larger, biologically significant phylogenies is an ongoing challenge in the field, the following results will serve as a foundational proof-of-concept rather than a replacement to current methods.

It is important to note that the RAxML trees are binary, except for the first element, where the root sometimes has three child nodes. Although this could potentially affect

[1] All the code and instances can be found in https://github.com/Nicofero/PhyloTree_ReConst.

the comparison values, it did not pose a problem, as the analysis remains the same in either case. Furthermore, the sequence length is not a parameter of the QNMcut method itself; it only influences the generation of the similarity matrix.

4.2 Robinson–Foulds distance to compare phylogenies

We will use one main metric to compare the generated phylogenetic trees with the base tree from RAxML, the *Robinson–Foulds* (RF) distance [21]. This distance can also be referred to as "symmetric difference," since it represents the non-trivial partitions that appear in one tree but not in both.

Consider two phylogenetic trees, T_1 and T_2, then the RF distance between the trees is defined as:

$$d(T_1, T_2) = i(T_1) + i(T_2) - 2v_s(T_1, T_2), \tag{6}$$

where $i(T_k)$ denotes the number of internal branches of the tree k and $v_s(T_1, T_2)$ represents the number of splits that exist in both trees. Although this distance is an absolute distance, its maximum value is reached when the two trees share no common splits, resulting in $v_s(T_1, T_2) = 0$. Thus, we can calculate the relative Robinson–Foulds distance as:

$$\mathrm{RF} = \frac{d(T_1, T_2)}{i(T_1) + i(T_2)}.$$

This relative distance can be interpreted as the percentage of error between the branches of two trees and takes values between 0 and 1. However, what we seek is the opposite, the percentage of correctly recovered branches, which leads to the performance metric used in the subsequent comparison:

$$\mathrm{CRB} = (1 - RF) \cdot 100. \tag{7}$$

To compute this value in Python, we used the ETE3 library [22], which allows the calculation of the RF distance between two trees (read from Newick files). Using the obtained RF distance, we can calculate the CRB for any pair of phylogenetic trees.

Alternative metrics, such as generalized RF distances, specifically Clustering Information Distance [23], were considered for their higher sensitivity to minor topological variations. Preliminary analyses yielded outcomes consistent with the standard Robinson–Foulds distance. Consequently, to facilitate a contextual alignment with [6], the RF distance (CRB metric) was prioritized. This consistency suggests that the observed performance is independent of the specific topological metric employed.

4.3 Quantum computing platforms

To design and execute the Quantum Annealing process, the Ocean library [24] for Python was used. Developed by D-Wave, this package provides the tools required to formulate Quantum Annealing problems and the capability to run them on actual quantum hardware. Using the functionalities offered by Ocean, we can easily define the formulation from Eq. 4 and execute it on the Advantage QPU 6.4 [25], a real quantum

Table 1 Performance comparison between QA, QAOA, and NJ (% of correctly recovered branches)

	QA	QAOA	NJ
Worst	10%	5%	20%
Best	47.83%	47.06%	100%
Average	32.95%	19.70%	76.97%

annealer designed to efficiently solve QA problems. To optimize for execution time, the experiment was conducted using a basic configuration of 64 reads per instance. This setup utilized the default 20 μs annealing schedule and a minor embedding to map the logical problem onto the physical hardware.

On the other hand, QAOA was executed on simulators, using both exact and noisy configurations. The execution utilized a single-layer circuit ($p = 1$), as increasing the depth to two or three layers yielded negligible performance gains in preliminary tests. The variational parameters were optimized using the COBYLA algorithm with a limit of 500 iterations, employing 1024 shots to ensure statistical reliability. To define and run the QAOA circuits, we used Qiskit 1.2.9 [26]. Qiskit Aer was employed to simulate the circuits, and the IBM Runtime was used to obtain the providers required for the noisy simulations. Due to limitations in the IBM Quantum Platform, we were unable to execute this method on IBM QPUs; however, fake providers enable us to approximate how the NMcutQAOA would behave on real quantum hardware. The algorithm was tested using two of IBM's fake providers, *FakeAlgiers* and *FakeGuadalupeV2*. These providers were selected because their high precision and relatively small number of qubits improve the efficiency of the simulations. In addition, we evaluated the performance of the Qmio quantum computer, installed at the Galicia Supercomputing Center (CESGA) [27], in comparison with IBM's QPUs by using the Qmio fake provider, *FakeQmio*. Qmio is a 32-qubit quantum computer located in CESGA, in Santiago de Compostela. Its QPU is based on coaxmon superconducting qubit technology.

5 Results

In this section, we present and discuss the obtained results using the experimental framework described in the previous section.

5.1 Comparison between methods

First, we compared both NMcutQA and simulated NMcutQAOA with another phylogeny reconstruction method. The method selected for comparison was the neighbor-joining (NJ) algorithm, in which the similarity matrix was first directly converted into a distance matrix and then used as input for the NJ implementation in Biopython [19]. The results are presented in Fig. 2, and a summary is provided in Table 1, which reports the best, worst, and average precision achieved by each method.

We can clearly observe that the NJ method achieves much better results than both graph-cutting methods. On average, the results obtained by the classical method

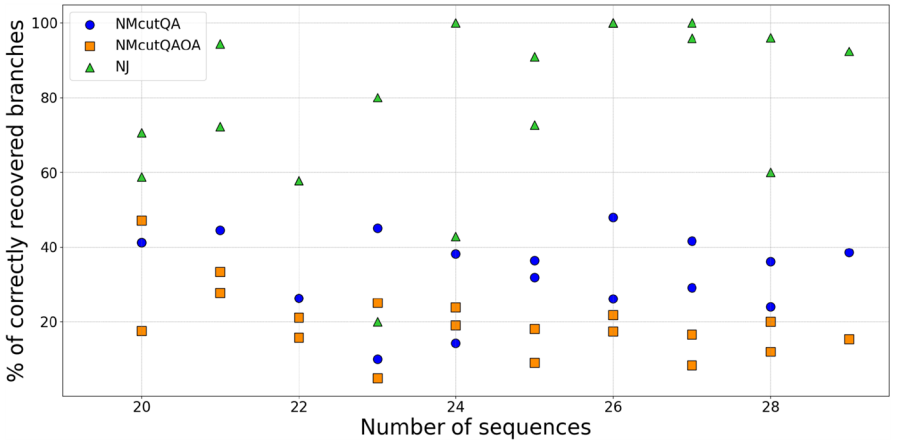

Fig. 2 Scatter plot representing a comparison between the NMcutQA, simulated NMcutQAOA, and neighbor-joining. The X axis represents the number of taxa used to generate the tree, and the Y axis represents the percentage of correctly recovered branches. We observe that the classical method yields better results than both quantum methods

represent an improvement of approximately 40% and 50% over QA and QAOA, respectively, indicating a clear performance advantage for the classical approach under the current experimental conditions. On the other hand, we observe that the QA approach outperforms QAOA. This result is not unexpected, as QAOA can be regarded as a discretized version of Quantum Annealing. In such cases, discretization typically leads to a degradation in solution quality, since the continuous optimization of Quantum Annealing is replaced by a finite number of variational steps. Consequently, the solutions obtained through QAOA are generally less accurate than those produced by QA, as shown in both Fig. 2 and Table 1.

5.2 Comparison with the exact NMcut method

To accurately quantify the quantum loss introduced by the QNMcut method, i.e., the loss in precision due to quantum error, we compared its results not only against the exact solution derived from the NMcut formulation, but also against a classical simulated annealing (NMcutSA) approach. To obtain this reference, we employed a brute-force solver within the Ocean framework [24] to find the exact solution of the minimum cut optimization problem, and used a highly optimized Ocean solver as well for the simulated annealing implementation. Due to the high computational cost associated with obtaining the exact solution, we limited our comparison to trees with sizes ranging from 20 to 25 taxa. The obtained results are displayed in Fig. 3. The average percentage of correctly recovered branches for the exact method is 57.47%, and the relative distance from the quantum and SA methods is shown in Table 2. Comparison with the precision achieved by the exact method indicates that the quantum loss is smaller than the difference observed between the quantum methods and the NJ method. This distance reflects the degradation in performance caused by approximation and quantum

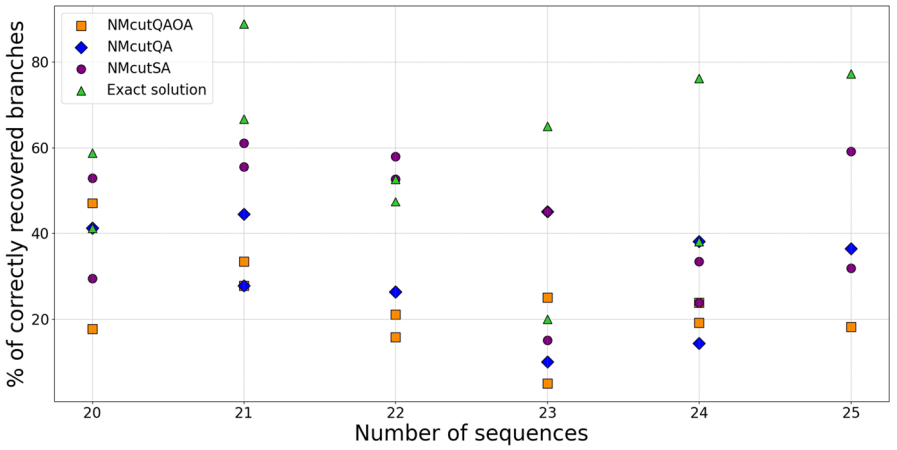

Fig. 3 Scatter plot representing a comparison between the NMcutQA, simulated NMcutQAOA, NMcutSA and exact NMcut method. The X axis represents the number of taxa used to generate the tree, and the Y axis represents the percentage of correctly recovered branches. The exact method obtains better results than the approximations, but not as precise as the neighbor-joining method represented in Fig. 2. On the other hand, SA obtains slightly better results than its quantum counterpart, in some cases achieving better results than the exact solution

Table 2 Relative distance between the optimal and the approximate solution for QA, QAOA, and SA. Negative numbers represent an improvement over the exact method.

	QA	QAOA	SA
Worst	62.49%	76.47%	56.25%
Best	0%	19.99%	−22.21%
Average	43.54%	59.10%	21.57%

variability. In this context, simulated annealing exhibits a smaller average distance to the exact solution than the quantum methods. Notably, there are cases where the heuristic solution outperforms the exact reference, meaning that optimized classic algorithms can obtain precise solutions to this problem.

These results suggest that the method may not be as precise as initially expected, indicating a potential need for variation or improvement in future iterations. Nevertheless, these findings provide valuable insight into the limitations and behavior of quantum approximations, guiding the direction for further development.

5.3 Comparison with QPU (noisy) QAOA

In this final QAOA comparison, we evaluated the precision obtained when running QAOA in noisy environments, thereby simulating the behavior of actual QPUs. To this end, as explained on Sect. 4.3, we employed fake providers, i.e., quantum backends designed to model the noise and imperfections in gate operations and measurements. These fake providers provide insight into how our circuit would perform on a real QPU, while offering a more accessible alternative for testing. Due to circuit constraints, this experiment was conducted exclusively on trees containing between 8 and 15

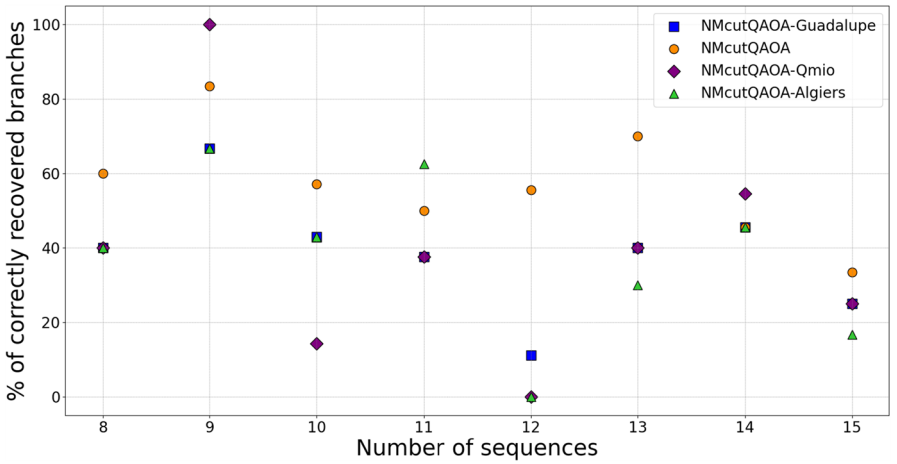

Fig. 4 Scatter plot representing a comparison between the exact and noisy simulations. The used fake providers are FakeGuadalupe, FakeAlgiers, and FakeQmio. The X axis represents the number of taxa used to generate the trees, and the Y axis represents the percentage of correctly recovered branches. On average, the exact simulation yields the best results (see Table 3); however, FakeQmio and FakeAlgiers achieve better solutions for certain sequence sizes

Table 3 Performance comparison in simulated QAOA between Aer, FakeGuadalupe, FakeQmio, and FakeAlgiers (% of correctly recovered branches)

	Aer	**FakeGuadalupe**	**FakeQmio**	**FakeAlgiers**
Worst	33.33%	11.11%	0.0%	0.0%
Best	83.33%	66.67%	100%	66.67%
Average	56.85%	38.57%	38.92%	38.02%

taxa. Using these types of trees implies that errors significantly reduce precision; however, it becomes easier to obtain a more precise tree since there are fewer possible configurations. The obtained results are shown in Fig. 4, and Table 3 presents the worst, best, and average precision for each simulation.

As expected, the exact simulation achieves the best results among the four methods. Interestingly, this comparison also reveals a phenomenon in which the introduction of noise leads to an improvement in the results. This phenomenon occurs because QAOA is a method that approximately explores the Hilbert space. The introduction of noise can lead to improvements in exploration and the discovery of better solutions. This is what happens, for example, in the tree of 9 sequences, where FakeQmio achieves a precision of 100% and reconstructs the tree perfectly. Moreover, FakeQmio slightly outperforms both IBM fake providers in both the best solution and average precision. However, a closer examination of the trees generated by Qmio highlights a limitation of the RF distance: all of the generated trees are pectinate, a consequence of the high operational error rate and the circuit depth (all-to-all interactions). Since we are only comparing the inner clades of the tree, the RF distance may register multiple coincidences, yet the overall tree topology is not accurately captured, in contrast

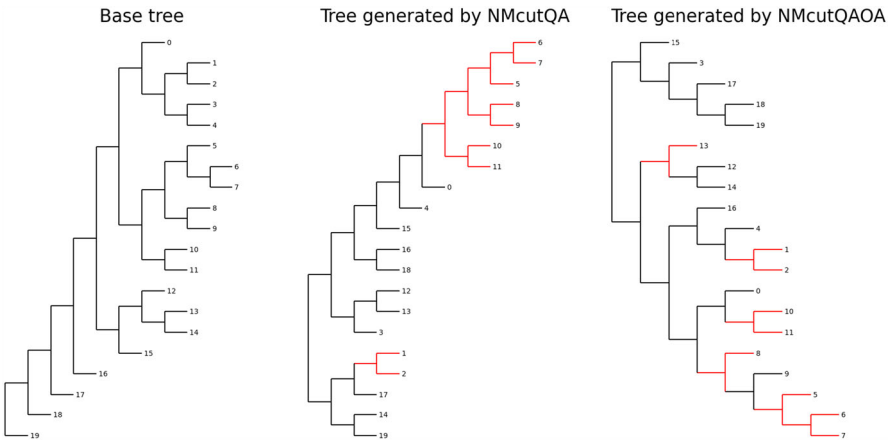

Fig. 5 Comparison between the trees generated by the NMcutQA and NMcutQAOA methods and the reference tree generated with RAxML. Correctly recovered branches between the reference and reconstructed trees are shown in red. Branch lengths are assumed to be constant

with the exact QAOA or QA approaches. This phenomenon also happens with the FakeAlgiers and FakeGuadalupe, although to a lesser extent.

These results highlight the main conclusion drawn from the previous sections: The QAOA method may not be the most suitable approach for this algorithm, as it yields poorer results that cannot be accurately reproduced on real quantum hardware.

5.4 Tree comparison

To conclude the comparison between methods, and before discussing the complexity of QNMcut, we present a qualitative visual analysis of the trees generated by both NMcutQA and NMcutQAOA. This comparison illustrates how the resulting trees differ from each other and how they relate to the original reference tree.

Figure 5 shows an illustrative example of this comparison using a tree with 20 taxa. Running this analysis across multiple trees, we observe that NMcutQA tends to recover more complete clades, while NMcutQAOA primarily recovers sparse internal branches. This qualitative observation indicates that, when the goal is to recover complete clades from a phylogenetic tree, NMcutQA is preferable to NMcutQAOA.

6 Experimental computational complexity

Next, we will evaluate the time required by each section of the method. First, we briefly discuss the generation of the similarity matrix, and then we analyze the computational complexity of the QNMcut method.

It is essential to note that generating the similarity matrix is a one-time process; it can be performed independently of the QNMcut method, and the resulting matrices

can be reused thereafter. The reconstruction of the similarity matrix is a deterministic method, and the complexity order can be easily theoretically defined as follows:

Proposition 1 *Given a MSA (or PSA) of n taxa, with a constant sequence length l, the generation of a similarity matrix D has a complexity order of* $\mathcal{O}\left(\frac{1}{4}n^2 l\right)$.

In contrast, QNMcut is not a deterministic method, and its computational complexity cannot be directly characterized by an exact order. However, we can analyze its upper and lower complexity bounds and validate the order empirically using experimental data. To do this, we first define the bounds of our problem. It is important to note that the complexity of the QNMcut method arises from solving the minimum cut optimization problems. As explained in Sect. 3.3, for each generated subgraph, we must solve a minimum cut problem for every possible cut configuration. The bounds for the number of optimization problems to be solved are given by Proposition 2.

Proposition 2 *Given a similarity matrix D of size n, the maximum and minimum number of minimum cut problems that need to be solved are obtained by the following expressions:*

$$Max_{prob} = \sum_{i=1}^{m} i = \frac{m(m+1)}{2}, \tag{8}$$

$$Min_{prob} = \lceil m \log_2 m \rceil, \tag{9}$$

where $m = \lfloor n/2 \rfloor$.

For the maximum case, the Max_{prob} number of problems is obtained when only one element is cut on each stage. Therefore, we will obtain that the maximum complexity order is:

$$\text{Max}_{\text{prob}} \sim \mathcal{O}(m^2).$$

Similarly, the Min_{prob} number of problems corresponds to the scenario in which every cut is chosen to be perfectly balanced, whenever such a division is possible. Therefore, the minimum complexity order is:

$$\text{Min}_{\text{prob}} \sim \mathcal{O}(m \log m).$$

As stated in [28], the complexity of quantum annealing is estimated to be $\mathcal{O}(e^{\sqrt{n}})$, representing an exponential improvement over the classical simulated annealing method ($\mathcal{O}(e^n)$). While this difference may indicate a possible theoretical advantage for quantum annealing in the context of the minimum cut problem as the number of instances increases, it remains an asymptotic estimate and should be interpreted cautiously, as it does not imply a demonstrated practical advantage over optimized classical solvers. However, given Proposition 2, one question remains: although we have defined bounds for the maximum and minimum number of problems, we have not provided a mean or expected value for that number. Deriving this theoretically is impossible, so we will attempt to estimate it experimentally.

To achieve this, we tested the problem for multiple sizes using a QA approach, obtaining the number of optimization problems, as shown in Table 4. Higher-taxa alignments were subsequently obtained using RAxML, as described in Sect. 4.1.

Table 4 Number of optimization problems needed to solve for variable sizes

Size	5	10	15	20	25	50	75
Optimization problems	10	13	25	56	102	148	257

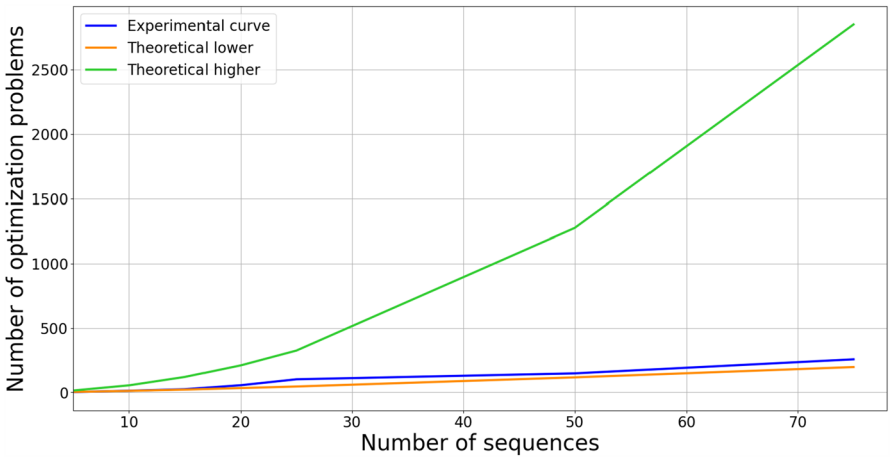

Fig. 6 Experimental curve for the number of optimization problems the QNMcut method needs to solve for each problem size. We can see that the curve follows a quasi-linear growth, very close to the lower bound

To understand how this evolution fits within the defined bounds, we graphically represented both the upper and lower bounds together with the theoretical curve, as shown in Fig. 6. Using this knowledge, we fitted a curve with SciPy [29] and found that the growth function can be approximated by the formula $f(x) = 0.35x \log(10.82x)$. We tested this hypothetical formula on all test instances described in the previous sections, as well as on a problem of size 100. The results showed good agreement, with errors below 10% in all cases. In most cases, the number of optimization problems was even lower than the predicted value.

7 Conclusions

This study aimed to evaluate the potential of quantum computing for applications in bioinformatics, specifically for the reconstruction of phylogenetic trees. We developed a quantum approach to phylogeny reconstruction based on a graph-splitting method, referred to as QNMcut. The algorithm operates by recursively partitioning the similarity matrix of the individuals, following a top-down approach to phylogenetic tree reconstruction. In our experiments, QNMcut achieved a level of precision comparable to its classical counterpart, as reported in [6] (approximately 40% precision).

The QNMcut algorithm was implemented using both Quantum Annealing (QA) and the Quantum Approximate Optimization Algorithm (QAOA) to address the underlying optimization problems. These approaches were evaluated in comparison with

established phylogeny reconstruction methods. It is important to note that the primary objective was to assess the feasibility of applying quantum computing to this problem, rather than to surpass the performance of existing classical approaches. As such, the limited tree sizes used here reflect current hardware constraints and are intended to establish a methodological foundation rather than a replacement for classical reconstruction. As expected, both quantum methods exhibited lower average performance compared to the classical methods, with QA achieving better results when executed on real quantum hardware. However, these lower results are attributable not to the quantum approach itself but to the NMcut method. Despite these lower accuracies, the NMcut framework remains a compelling candidate for quantum adaptation due to its performance characteristics in specific evolutionary scenarios. As noted in [6], the NMcut method tends to outperform alternative phylogeny reconstruction methods as sequences diverge (higher branch lengths). Although the data utilized in this study did not provide the high branch lengths necessary to demonstrate this specific advantage, QNMcut establishes the methodological groundwork for a tool potentially superior in high-divergence evolutionary modeling. This suggests that future work should focus on exploring alternative reconstruction strategies rather than solely seeking higher precision within the current framework.

Our results indicate that quantum methods for solving the underlying optimization problems may offer potential computational advantages over classical approaches; however, this expectation is based primarily on theoretical complexity considerations. Consequently, any such advantage should be interpreted as prospective rather than as a demonstrated practical improvement. However, it also reveals a fundamental limitation of the NMcut method: the rapid growth in the number of optimization subproblems required to reconstruct a complete tree. This limitation is particularly pronounced when the trees are inherently pectinate and is further exacerbated if the solver tends to produce pectinate topologies, as observed with QAOA on real QPUs.

Future work will focus on developing alternative methods that leverage quantum optimization for phylogeny reconstruction. These could include variations of neighbor-joining, maximum parsimony, or clustering approaches, potentially achieving higher precision while retaining the advantages of the current method. Further research should also explore improved quantum solvers and configurations to enhance both the accuracy and quality of the reconstructed trees.

Acknowledgements The authors acknowledge CESGA (Centro de Supercomputación de Galicia) for providing access to the QMIO quantum computer. This work has received financial support from the Agencia Estatal de Investigación (Spain) (PID2022-141623NB-I00 and PID2022-137061OB-C22), Xunta de Galicia - Consellería de Cultura, Educación, Formación Profesional e Universidades (Centro de investigación de Galicia accreditation 2024-2027 ED431G-2023/04 and Reference Competitive Group accreditation ED431C-2022/016), and the European Union (European Regional Development Fund - ERDF).

Author Contributions N.F.O. developed the software, conducted all experiments, and drafted the manuscript. All authors contributed to the design of the research idea and experimental framework. P.F.P. and J.C.P. critically reviewed and revised the manuscript. All authors read and approved the final version of the manuscript.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data Availability Sequence data and ground-truth phylogenetic trees have been extracted from RAxML Grove (https://github.com/angtft/RAxMLGrove)

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Felsenstein J (2004) Inferring Phylogenies. Sinauer Associates, Sunderland, Massachusetts
2. Saitou N, Nei M (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol 4(4):406–425. https://doi.org/10.1093/oxfordjournals.molbev.a040454
3. Warnow T (2017) Computational Phylogenetics: An Introduction to Designing Methods for Phylogeny Estimation. Cambridge University Press, Cambridge
4. Sokal RR, Michener CD (1958) A statistical method for evaluating systematic relationships. University of Kansas Scientific Bulletin 38:1409–1438
5. Zhang S-B, Zhou S-Y, He J-G, Lai J-H (2011) Phylogeny inference based on spectral graph clustering. J Comput Biol 18(4):627–637
6. Onodera W, Hara N, Aoki S, Asahi T, Sawamura N (2023) Phylogenetic tree reconstruction via graph cut presented using a quantum-inspired computer. Mol Phylogenet Evol 178:107636. https://doi.org/10.1016/j.ympev.2022.107636
7. Aramon M, Rosenberg G, Valiante E, Miyazawa T, Tamura H, Katzgraber HG (2019) Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. Front Physi. https://doi.org/10.3389/fphy.2019.00048
8. Kadowaki T, Nishimori H (1998) Quantum annealing in the transverse Ising model. Phys Rev E 58(5):5355–5363. https://doi.org/10.1103/PhysRevE.58.5355
9. Farhi E, Goldstone J, Gutmann S (2014) A Quantum Approximate Optimization Algorithm. https://doi.org/10.48550/arXiv.1411.4028
10. Sievers F, Higgins DG (2014) In: Russell, D.J. (ed.) Clustal Omega, Accurate Alignment of Very Large Numbers of Sequences, pp. 105–116. Humana Press, Totowa, NJ . https://doi.org/10.1007/978-1-62703-646-7_6
11. Price MN, Dehal PS, Arkin AP (2010) Fasttree 2 – approximately maximum-likelihood trees for large alignments. PLoS ONE 5(3):1–10. https://doi.org/10.1371/journal.pone.0009490
12. Stamatakis A (2014) Raxml version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics 30(9):1312–1313. https://doi.org/10.1093/bioinformatics/btu033
13. Piñeiro C, Pichel JC (2024) Efficient phylogenetic tree inference for massive taxonomic datasets: Harnessing the power of a server to analyze 1 million taxa. GigaScience. p 55. https://doi.org/10.1093/gigascience/giae055
14. Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22(8):888–905. https://doi.org/10.1109/34.868688
15. Matsui M, Iwasaki W (2019) Graph splitting: A graph-based approach for superfamily-scale phylogenetic tree reconstruction. Syst Biol 69(2):265–279. https://doi.org/10.1093/sysbio/syz049
16. Combarro EF, Gonzalez-Castillo S (2023) A Practical Guide to Quantum Machine Learning and Quantum Optimisation. Packt Publishing, Birmingham, UK
17. Henikoff S, Henikoff JG (1992) Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci U S A 89(22):10915–10919. https://doi.org/10.1073/pnas.89.22.10915

18. Dufour YS, Kiley PJ, Donohue TJ (2010) Reconstruction of the core and extended regulons of global transcription factors. PLoS Genet 6(7):1–20. https://doi.org/10.1371/journal.pgen.1001027
19. Cock PJA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B, Hoon MJL (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics 25(11):1422–1423. https://doi.org/10.1093/bioinformatics/btp163
20. Höhler D, Pfeiffer W, Ioannidis V, Stockinger H, Stamatakis A (2021) Raxml grove: an empirical phylogenetic tree database. Bioinformatics 38(6):1741–1742. https://doi.org/10.1093/bioinformatics/btab863
21. Robinson DF, Foulds LR (1981) Comparison of phylogenetic trees. Math Biosci 53(1):131–147. https://doi.org/10.1016/0025-5564(81)90043-2
22. Huerta-Cepas J, Serra F, Bork P (2016) Ete 3: reconstruction, analysis, and visualization of phylogenomic data. Mol Biol Evol 33(6):1635–1638. https://doi.org/10.1093/molbev/msw046
23. Smith MR (2020) Information theoretic generalized robinson-foulds metrics for comparing phylogenetic trees. Bioinformatics 36(20):5007–5013
24. D-Wave Systems Inc.: D-Wave Ocean Software Documentation. (2018). Accessed: 2025-01-15. https://docs.ocean.dwavesys.com/projects/system/en/stable/reference/
25. McGeoch C, Farré P (2022) Advantage processor overview. Technical report, D-Wave Systems Inc
26. Javadi-Abhari A, Treinish M, Krsulich K, Wood CJ, Lishman J, Gacon J, Martiel S, Nation PD, Bishop LS, Cross AW, Johnson BR, Gambetta JM (2024) Quantum computing with Qiskit .https://doi.org/10.48550/arXiv.2405.08810
27. Galicia Supercomputing Center (CESGA): Qmio: The Galician quantum computer. Accessed: 2025-10-08 (2023). https://quantum.cesga.es/infraestructures/
28. Mukherjee S, Chakrabarti BK (2015) Multivariable optimization: Quantum annealing and computation. The Euro Phys J Special Topics 224(1):17–24. https://doi.org/10.1140/epjst/e2015-02339-y
29. ...Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P (2020) SciPy 1.0 Contributors: SciPy 1.0: fundamental algorithms for scientific computing in python. Nat Methods 17:261–272. https://doi.org/10.1038/s41592-019-0686-2