Quantum computing: a contribution to accelerate and enhance multiple sequence alignment

An investigation by **NTT DATA** examines how Quantum Annealer technology can tackle problem-solving related to multiple sequence alignment (MSA), a crucial field within bioinformatics.

1. Executive summary

Quantum computing, a rapidly advancing field of technology, holds the promise of solving complex problems more faster than classical computing—sometimes even at an exponential acceleration.

In this particular technical study, **NTT DATA** harnesses Quantum Annealer technology to expedite the resolution of challenges linked to multiple sequence alignment (MSA), a critical task in the domain of bioinformatics crucial for representing and comparing biological sequences (DNA, RNA, or primary protein structures).

Comparative performance with classical approaches like Clustal W and MUSCLE is demonstrated, thereby laying the groundwork to bolster ongoing explorations in this dynamic field.

2. Introduction

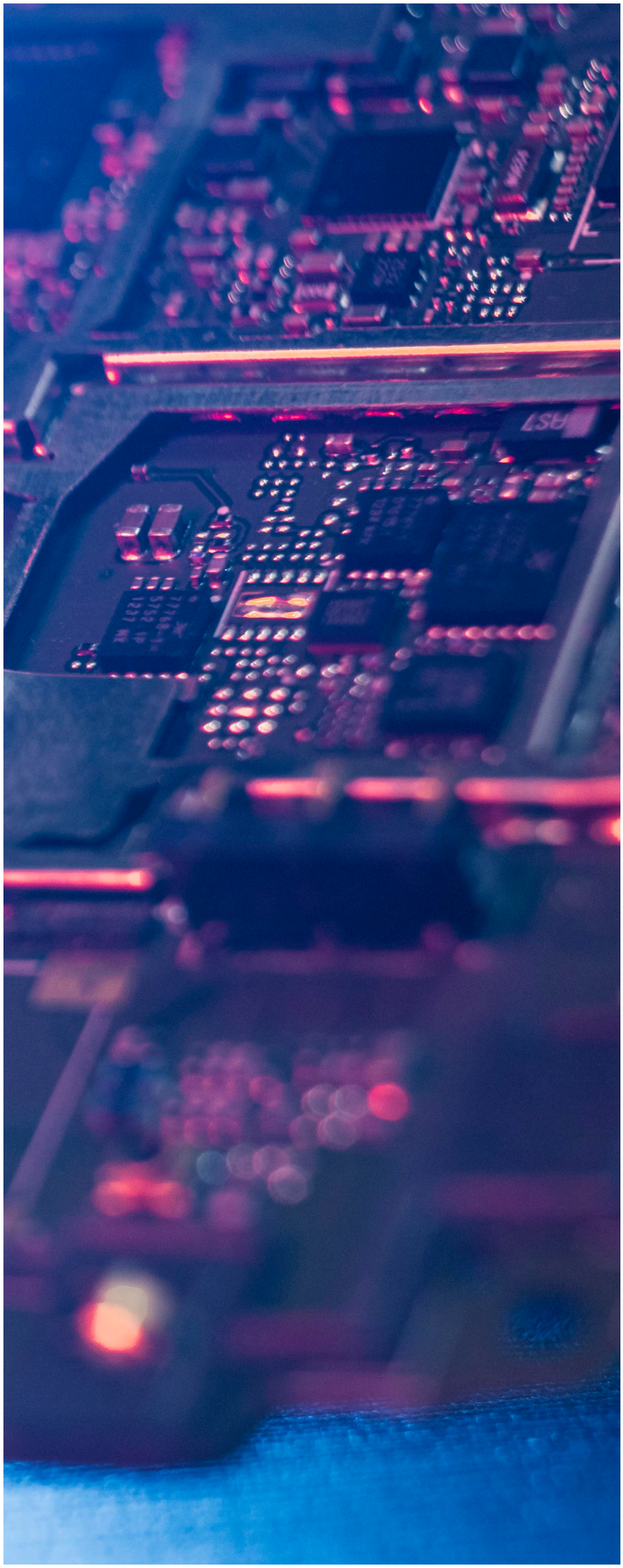
Multiple Sequence Alignment (MSA), a cornerstone in bioinformatics, plays a fundamental role in comparing biological sequences by aligning them following a specific scoring scheme. MSA primarily serves to decipher sequence homology and unravel phylogenetic relationships, thereby illuminating the evolutionary trajectories of species [1]. Despite its significance, MSA poses substantial computational challenges. It is generally classified under the category of non-deterministic polynomial time hardness (NP-hard), implying that no existing algorithm can find an optimal solution within polynomial time on a deterministic Turing machine [2, 3].

Historically, the foundation in sequence alignment was established by the algorithms of Needleman-Wunsch (1970) and Smith-Waterman (1981) [4, 5]. Although they were designed for pairwise alignments, these algorithms were essential in laying the groundwork for MSA [6]. Subsequent advancements led to

heuristic-based algorithms like Clustal and Multiple Sequence Comparison by Log-Expectation (MUSCLE), which improved alignment efficiency for larger data sets [7, 8]. However, these methods often face increasing computational complexity as the number of sequences grows, a limitation highlighted in recent comparative studies [9]. These hindrances have allowed the exploration of alternative approaches, including machine learning techniques and, more recently, quantum computing [10].

In the rapidly advancing field of quantum computing, there is an expectation that many problems will be solved faster than by classical computers, potentially offering exponential speedup in certain cases [11]. Considering the current constraints in gate-based quantum computing, particularly in terms of qubit count, gate fidelity, and circuit depth, quantum annealing stands out as a practical alternative as it provides an infrastructure capable of encoding bigger problems. Accordingly, in this paper, MSA is modeled as a Quadratic-Unconstrained-Binary Optimization (QUBO) formulation, originally proposed in [12]. For our convenience, QUBO formulations can be naturally encoded into Quantum Annealers.

We used Dwave's Quantum Annealer to find solutions to the proposed QUBO and compared its performance and run-time against classical approaches, namely Clustal W and MUSCLE. Our findings seek to deepen the understanding of quantum annealing's role in resolving MSA challenges, and test weather quantum Annealers can provide any form of computational speedup at all, thereby setting the foundation for ongoing explorations in this dynamic and seemingly promising field.



3. Multiple Sequence Alignment

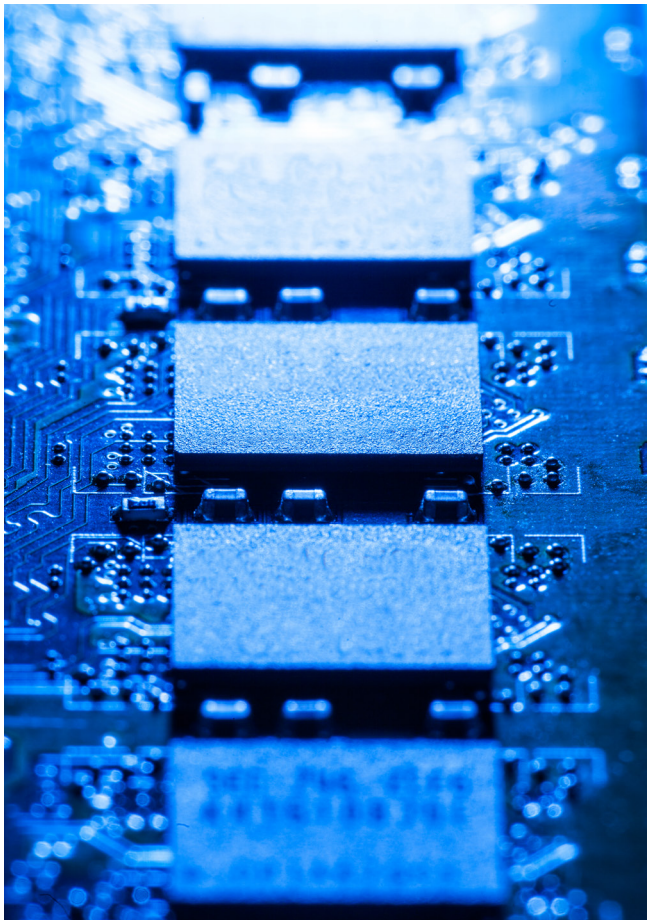
Recent advances in computing power and improved sequencing techniques have allowed researchers to align longer DNA sequences more accurately [13, 14, 15]. Moreover, there has been an explosion in the availability of genomics data by many orders of magnitude than decades ago, thanks to Next-Generation sequencing technology [16, 17]. Therefore, it is now possible to perform sequence alignment of more sequences of greater length, yet algorithms are not getting substantially faster. There exist two types of sequence alignments, pairwise sequence alignment, and multiple sequence alignment. The first is an already solved problem in the domain of computer science, meaning, an exact solution can be found within polynomial time, using algorithms like Needlemann-Wunsch or Smith-Waterman [18, 19]. The latter is unsolved and belongs to a subset class called NP-complete problems, which implies that a global optimal solution cannot be found in polynomial time [2]. It is therefore important to find more efficient ways to tackle MSA, as improved alignments could pave the way to big technological breakthroughs in areas such as phylogeny inference, protein structure and functional analysis [20, 21].

3.1 Problem Outline

Pairwise sequence alignment serves as the basis of multiple sequence alignment, which task is to insert gaps into the shortest sequence to improve the vertical agreement between each column. Figure 1 depicts an example of the alignment of two sequences.

Figure 1: A possible pairwise alignment of two sequences.

Multiple sequence alignment refers to the process of aligning three or more biological sequences and consists of two variations, one named global sequence alignment (GA) and the other local sequence alignment (LA). The first consists of aligning the entire composition of biological sequences, and the latter consists of aligning solely a specific area within the sequences. Both are important and used for different purposes.



3.2 Mathematical formulation of MSA

Consider a set of N biological sequences $S = \{s_i\}^N_{i=1}$ where $N \geq 3$ and each sequence si is composed only of the elements: {A, C, T, G}. Let $l_i = |s_i|$ denote the length of the ith sequence, and $L = \max(l_i)$ the length of the longest (reference) sequence. Now, let $s_{i,j}$ denote the j th character of sequence i, where $1 \leq j \leq li$. This information can be represented as the following mathematical expression:

$$S = \begin{bmatrix} s_1 = (s_{1,1}, s_{1,2}, \ldots, s_{1,l_1}) \\ s_2 = (s_{2,1}, s_{2,2}, \ldots, s_{2,l_2}) \\ \vdots \\ s_N = (s_{N,1}, s_{N,2}, \ldots, s_{N,l_2}) \end{bmatrix} \quad \textbf{e.g. } S = \begin{bmatrix} \mathbf{s_1 = (C, C, T, G)} \\ \mathbf{s_2 = (C, G)} \\ \vdots \\ s_N = (C, T, G) \end{bmatrix}$$

One possible variation of MSA is to assume that, the sequences will be filled with gaps such that all match the size of the longest sequence (i.e. all have a size L). This is the case analyzed in the present work. The task at hand is thus to re-accommodate every sequence in S to achieve maximum vertical similarity.

3.3 Scoring Strategy

There are multiple ways of evaluating the quality of an aligned set of sequences, each with a different scoring scheme. Additionally, the alignment scheme depends on the intended purpose of the alignment. The Sum-of-Pairs (SP) scoring scheme is a common scheme that assigns a scoring based on the degree of vertical similarity between the sequences, post alignment. Given a set of aligned sequences, the Sum-of-Pairs is defined as the sum of all distinct pairwise comparisons in a column, over all columns:

$$\text{Minimize} \sum_{j=1}^{N} \sum_{j'=j+1}^{N} \sum_{i=1}^{L} \text{simSP}(s_{j',i}, S_{j',i}) \quad (1)$$

Note that $j' = j + 1$, meaning sj' is the next sequence after sj .
The scoring scheme is defined as follows:

$$\text{simSP}(_{sj,i,\ sj',i}) = \begin{cases} -1, & s_{j,i} = s_{j',i} \\ +1, & s_{j,i} \neq s_{j',i} \\ 0, & s_{j,i} = \text{`}-\text{'} \vee sj',i = \text{`}-\text{'} \end{cases} \quad (2)$$

Note that, a match gives a −1 value, a mismatch throws a positive value, and a gap returns null. It is therefore desired to *minimize* the objective function once set. This scoring scheme results highly convenient given that the purpose of quantum annealing is to find the global minimum energy value of a hamiltonian.

3.4 Computational Complexity of MSA

The computational complexity of MSA depends on the particular approach used to solve it. In some cases, the resulting alignment can have more columns than the length of the original reference sequence. In the scenario under consideration, sequences are aligned in relation to a reference sequence with a length of l_M , and its assumed that $l_i < l_M \ \forall_i$, meaning, the task is to fill si with gaps such that $l_i = l_M \ \forall_i$. That stated, a sequence li requires $l_M - l_i$ gaps to complete the entire sequence, in which case every gap can occupy any one of l_M positions. As such, the number of possible gap configurations a single sequence can have with respect to the reference sequence is given by the binomial coefficient:

$$\begin{bmatrix} l_M \\ l_M - l_i \end{bmatrix}, \forall i \quad (3)$$

Now, given a set of N sequences including the reference, the task is to align $N - 1$ sequences to that reference sequence. The number of possible gap configurations is represented by S, mathematically described by:

$$S = \prod_{i-1}^{N-1} \begin{bmatrix} l_M \\ l_M - l_i \end{bmatrix} \quad (4)$$

As evident from 4, MSA is affected by the 'curse of dimensionality', with the space of potential gap configurations expanding approximately exponentially as additional sequences are incorporated. Considering a set of $N = 10$ sequences, where the longest sequence has a length of 40 and the rest of the sequences have a length of 30, the number of possible gap insertions becomes:

$$\begin{bmatrix} 40 \\ 10 \end{bmatrix}^9 \sim 2.25 \times 10^{80}$$

which is greater than the number of atoms in the visible universe. Exploring such a massive universe of possibilities becomes untraceable for conventional computers, which is why current methods rely on heuristics. Note in the previous analysis we are not analyzing the potential *letter* configuration arrangements. This is because analysing letter configurations introduces the possibility of unordered sequences, thus extending the configuration space to undesired cases. Thus, we are better off analysing plausible gap insertions.

4. Tested Algorithms

In the present section we provide a brief explanation on the different algorithms tested. It is important to point out that all of them are based on heuristics, meaning, none are guaranteed to provide an optimal solution.

4.1 ClustalW

ClustalW, an extension of the original Clustal series, adopts a progressive alignment strategy. It initially performs pairwise alignments and then builds a guide tree to align these pairs hierarchically. Key enhancements in ClustalW include sequence weighting and both general and position-specific gap penalties, which collectively improve alignment accuracy for sequences with significant divergence and mitigate biases from over-represented sequences [22]. For nucleotide sequences, ClustalW adopts a match/mismatch scoring system. This system assigns positive scores for nucleotide matches, incentivizing alignments of identical bases due to their evolutionary conservation. Mismatches are penalized with negative or lower positive scores, reflecting their lower likelihood of conservation [23].

In terms of computational complexity, Clustal W's operations can be divided into three stages. The first stage involves all-against-all pairwise alignments for *n* sequences, amounting to $\binom{n(n-1)}{2}$ pairwise comparisons. Given the time complexity of $O(L^2)$ for each pairwise alignment of sequences of length L, the resulting complexity is of $O(n^2L^2)$. For the second step, NJ or UPGMA is commonly used for the construction of the guide tree. In this case, the complexity can vary between $O(n^2)$ and $O(n^3)$, respectively. Finally, the progressive alignment incurs a complexity of approximately $O(n^2L^2)$

in the worst-case scenario. Consequently, the overall complexity of ClustalW is predominantly influenced by the quadratic relationship of the number of sequences and their lengths [24].

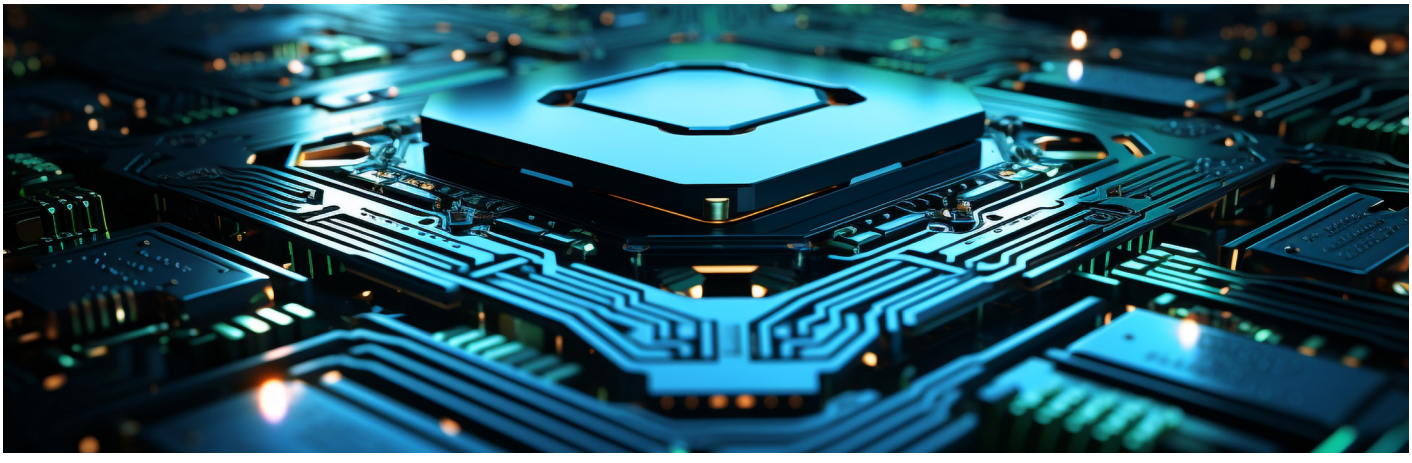
4.2 MUSCLE

The Multiple Sequence Comparison by Log-Expectation (MUSCLE) algorithm excels in executing large-scale align-ments, noted for its rapid and efficient processing [25]. Its core is an iterative refinement process, systematically improving the consensus alignment. Initially, MUSCLE generates a draft alignment via progressive alignment, forming a guide tree from a sequence-derived distance matrix, utilizing k-mer counting for sequence similarity assessment. In its subsequent phase, MUSCLE refines this alignment by modifying the guide tree and realigning sequences accordingly. The iterative refinement phase then adjusts alignment segments through tree-dependent restricted partitioning, culminat-ing in the final alignment. This phase critically involves Sum-of-Pairs (SP) scoring, aggregating pairwise scores for a comprehensive alignment evaluation [26].

The algorithm's computational complexity, considering its multi-stage nature, resists simple encapsulation. Typically, in large data sets with lengthy sequences, time complexity may reach $O(n^2L^2)$, with n as the sequence count and L as the average sequence length [26]. Space complexity is also notable, necessitating storage of distance matrices, guide trees, and alignments, indicative of MUSCLE's sophisticated design for handling complex bioinformatics alignment tasks [25].

4.3 Simulated Annealing

Simulated Annealing is a meta-heuristic technique for *approximating* global optimum values of an objective function. It is commonly used to find approximations to discrete optimization problems like the boolean satisfiability problem or the travelling salesman problem. The algorithm works in the following way. In every iteration, it evaluates a neighboring state s* relative to it's current state s. It then makes a probabilistic decision to either transition the system to state *s** or maintain it in state s. These probability-driven decisions are designed to guide the state towards the global optimal value. The process is repeated until a state is reached that is satisfactory for the intended application, or until the computational resources get exhausted. One problem with simulated annealing is that, through the process of iterating it comes across local minima, and may become trapped in them. Stochastic tunneling as well as quantum annealing attempt to overcome this problem, escaping the local minima in search for the global minima.

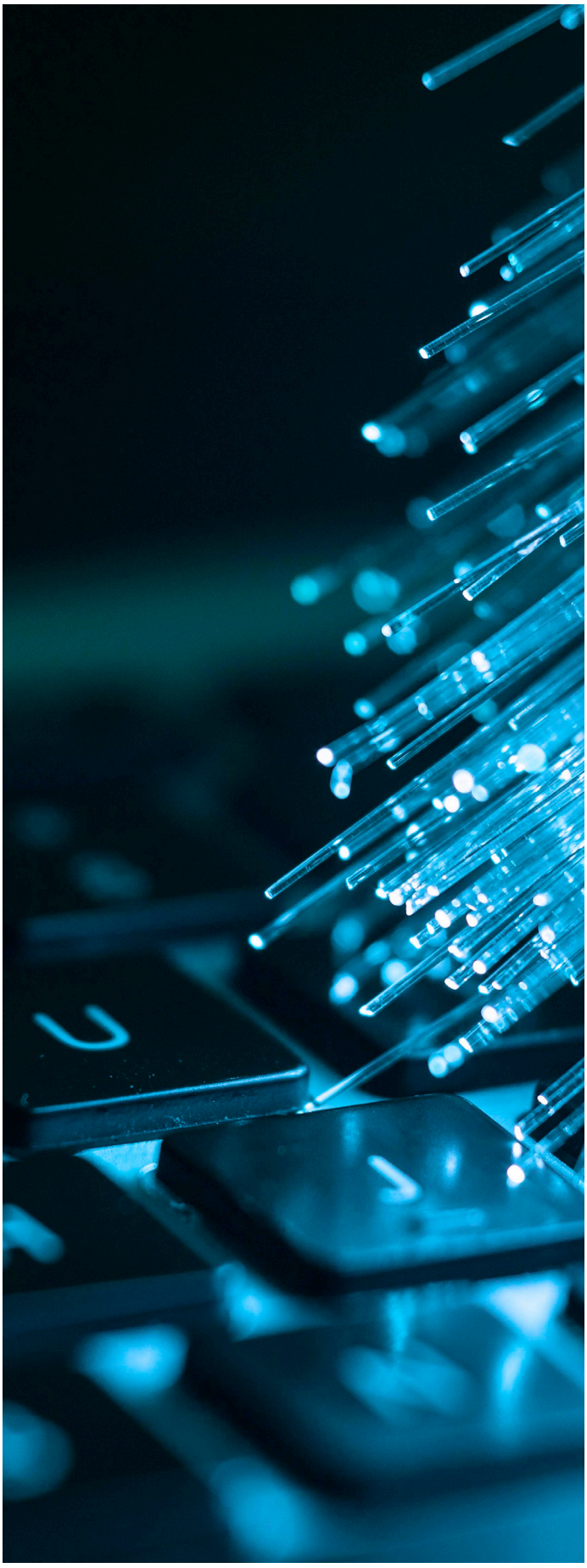


4.4 Quantum Annealing

Quantum Annealing (QA) is regarded as the analog version of quantum computing and leverages fundamental quantum mechanical properties, namely, quantum tunneling, quantum parallelism, the adiabatic theorem, and quantum fluctuations. Quantum Annealers seek to find the global minimum value or *ground-state* of a Hamiltonian, and are particularly suitable for finding the ground-state of a generic Ising Model. A reason why QA has drawn so much attraction is because important combinatorial optimization problems relevant to real-world scenarios like the traveling salesman problem, can be mapped into the Ising Model, which can be exploited using Quantum Annealing. The procedure goes as follows: the optimization problem is first mapped to an equivalent Ising model whose Hamiltonian, H_P , is known. It is assumed that HP is described by terms of the *z* components of the Pauli operator σ^z. Second, a driver Hamiltonian HD is introduced, which is not commutative with H_P , and has a trivial ground-state [27] [28]. The total system Hamiltonian is given by:

$$H(t) = A(t)H_D + B(t)H_P$$

It follows that *A(t)* and *B(t)* are functions satisfying A(ti) >> B(ti) and B(t_f) >> A(t_f) such that H(t) starts at the ground-state of H_D at t_i and ends at the ground-state of H_P at t_f . Initially, H(ti) ≈ A(t)HD. If the change in H(t) is gradual enough, the quantum system (i.e. the Ising model) obeys the adiabatic theorem of quantum mechanics, which states that a slow enough evolving quantum mechanical system should remain in the ground-state at any time t, following the time-dependent Schrödinger equation. Given that H_D becomes negligible at t_f, the quantum state should end up in the ground-state of the problem's Hamiltonian, HP . As opposed to the *Simulated Annealing approach,* Quantum Annealing uses *quantum fluctuations* instead of thermal fluctuations, allowing to cut through energy barriers due to *Quantum Tunneling* [27] [28], allowing to reach the ground state.



5. Mathematical Model

Given the nature of how quantum annealing works, the idea is to formulate it as a *Quadratic-Unconstrained-Binary-Optimization function* (QUBOs for short), which can then be easily mapped to an equivalent Ising model. QUBOs are expressed in vector form as the following:

$$x^T Qx + Bx, x \in \{0, 1\}nx1, Q \in \mathbb{R}^{nxn}, B \in \mathbb{R}^{1xn}$$

Essentially, QUBO formulations consist of a summation of quadratic and linear interactions between the variables.

5.1 Variable encoding

Firstly, we define $x^{s,n,i}$ as the decision binary variable, which represents whether the *n*th character of the *s*th sequence should be placed in the ith position $x^{s,n,i} = 1$, or not $x^{s,n,i} = 0$. The variable i spans the length of the longest sequence (recall sequences are being aligned with respect to the longest sequence). As it stands, this encoding includes cases where one letter can be placed in the same position as another letter within the same sequence (e.g. $x_{1,1,3} \equiv x_{1,2,3} \equiv 1$), it allows permutations in the order of the sequences (e.g. $x_{1,3,2} \equiv x_{1,2,3} = 1$), and also allows one nth character in a sequence to be placed in more than one i position. These issues are addressed by introducing penalty functions in subsequent sections. Every letter in s_i, can be in *L* number of positions. Analysing the case for all sequences, we obtain the number of binary variables:

$$n_q = L \sum_{i=1}^{N} li, \qquad (5)$$

where n_q represents the number of qubits required. This implies that the number of qubits required scales roughly polynomically with the length of the longest sequence: $O(L^2)$

5.2 Cost Function

In order to maximize vertical similarity, the scoring metric established in 6 is introduced. The procedure in establishing the cost function goes as follows. Essentially, every variable $x_{s,n,i}$ is compared with the variable $x_{s',n',i,}$ which represents a letter of the next sequence being in the ith position (or not). s′ = s + 1 and n′ is a letter belonging to sequence s′. For every potential vertical alignment between xs,n,i and $x_{s',n',i,}$ we analyze whether the letters produce a match, mismatch, or gap through the use of a rank 4 tensor defined as:

$$w_{s,n,s',n'} = \begin{cases} -1, s_n = s'_{n'} \\ +1, s_n \neq s'_{n'} \\ 0, s_n = \text{`-'} \vee s'_{n'} = \text{`-'} \end{cases} \quad (6)$$

Having defined our rank 4 tensor, the cost function C(x) is defined as the following:

$$C(x) = \sum_{s} \sum_{n'} \sum_{n} \sum_{i} \omega_{s,n,s',n'} x_{s,n,i} x_{s',n',i,} \quad (7)$$

where x comprises all the $x_{s,n,i}$ binary decision variables. Notice that, n′ is not necessarily equal to n + 1, but instead an independent variable.

5.3 Hard Constraints

It is imperative that constraints are introduced, given that the function as it stands will inevitably produce unfeasible solutions, due to the factors previously mentioned. As such, the following constraints are introduced:

Any nth letter of a string s must occur at exactly one ith of the L available columns:

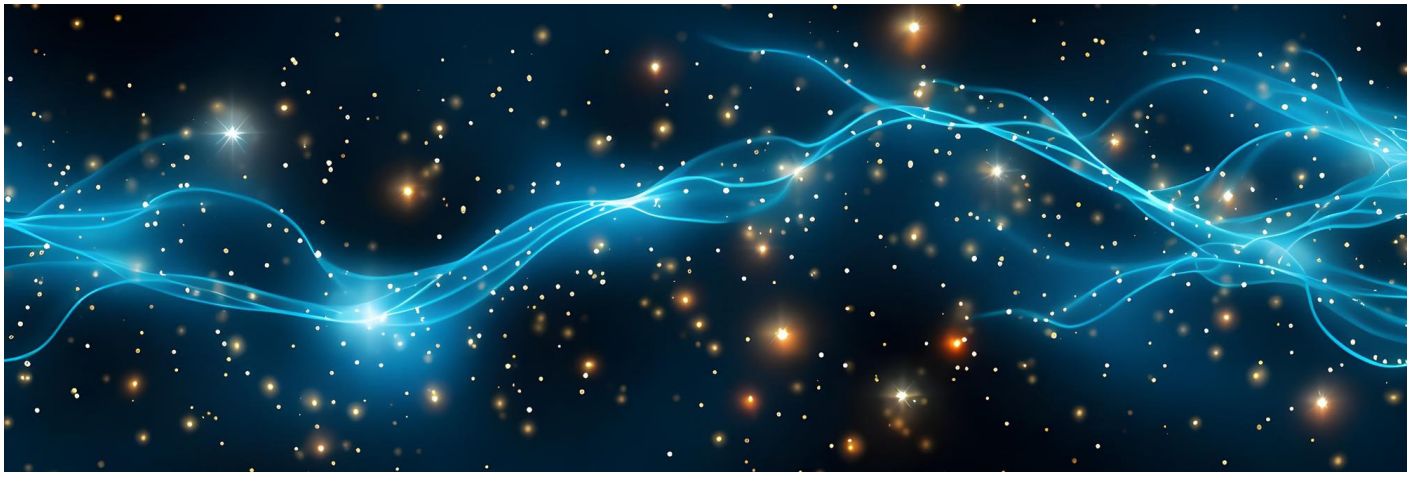
$$C(x) = \sum_{I=1}^{L} x_{s,n,i} = 1, \forall\, s, n$$

Any *i*th of the *L* available columns, in each row, must be occupied by one letter at most:

$$\sum_{n=1}^{L} {}^{S} x_{s,n,i} \leq 1, \forall\, s, n$$

Finally, the order of the sequence must be preserved. Put mathematically, no (*n* + 1)th letter can be placed before the *n*th:

$$\sum_{s=1}^{L} {}^{S} x_{s,n',i} \cdot x_{s,n,i'} = 0, \forall n < n', \forall i < i'.$$



5.4 Soft Constraints

It is desired to construct the objective function from the constraints previously mentioned. However, as the name QUBO suggests, the formulation does not accept constraints. To fix this, the hard constraints are mapped into soft constraints or penalty functions. These functions are quadratic-linear interactions which penalize unfeasible solutions. The degree of penalization is dictated by a coefficient called the Lagrange Multiplier. As such, the constraints are mapped into the following:

$$\sum_{i} x_{s,n,i} = 1, \forall\, s,n \longrightarrow p1 \sum_{s}\sum_{n}\left(\sum_{i} x_{s,n,i} - 1\right) \quad (8)$$

$$\sum_{n} x_{s,n,i} \leq 1, \forall\, s,i \longrightarrow p2 \sum_{s}\sum_{i}\sum_{n<n'} x_{s,n,i}\, x_{s,n',i} \quad (9)$$

$$\sum_{s} x_{s,n',i} x_{s,n,i'} = 0, \forall\, n < n', \forall\, i < i' \longrightarrow p3 \sum_{s}\sum_{n<n'}\sum_{i<i'} x_{s,n',i} x_{s,n,i'} \quad (10)$$

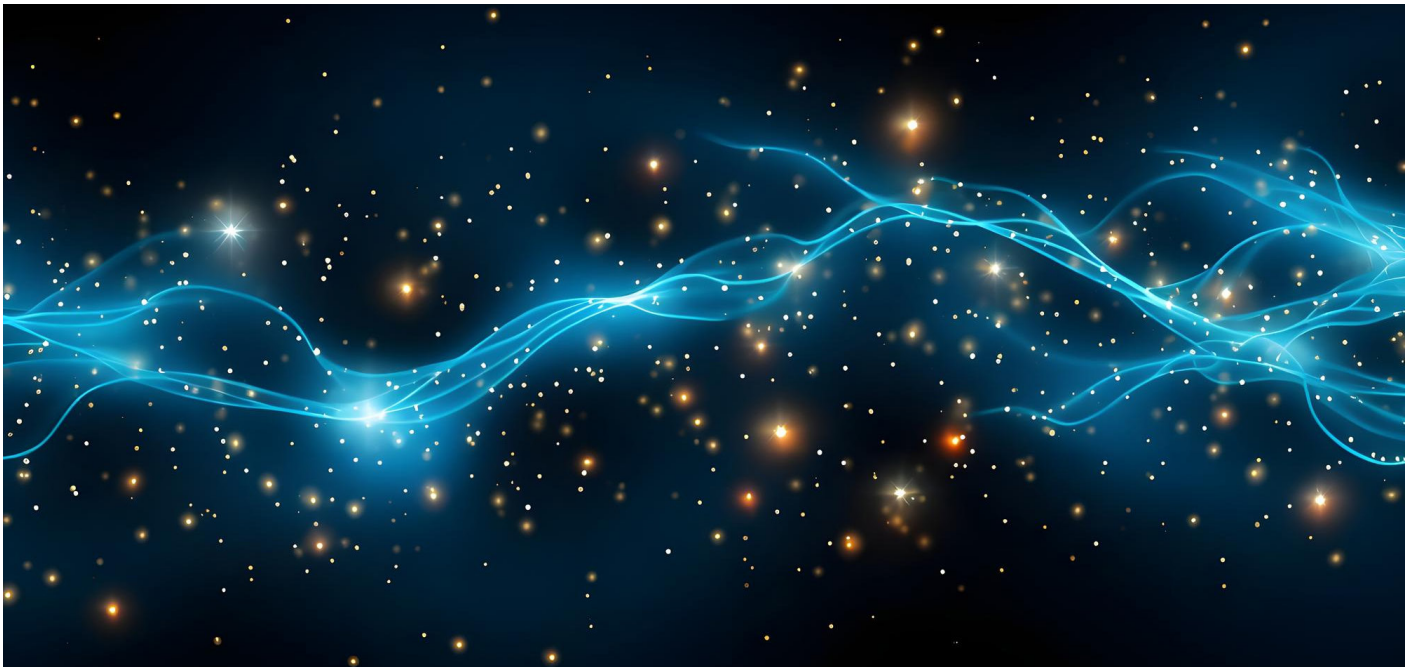
where *p1*, *p2* and *p3* are the Lagrange Multipliers for 8, 9 and 10, respectively. From 7, 8, 9, and 10, the following cost function is derived:

$$C(x) = \sum_{s}\sum_{n'}\sum_{n}\sum_{i} \omega_{s,n,s',n'} x_{s,n,i} x_{s',n',i} + p_1 \sum_{s}\sum_{n}\left(\sum_{i} x_{s,n,i} - 1\right)^2$$

$$+ p2 \sum_{s}\sum_{i}\sum_{n<n'}\sum_{i} x_{s,n,i} x_{s,n',i} + p_3 \sum_{s}\sum_{n<n'}\sum_{i<i'} x_{s,n',i} x_{s,n,i'}$$

where $x \in \{0, 1\}^{n}q$. Although not immediately obvious, C(x) is nothing more than a QUBO formulation, composed of linear and quadratic interactions which can be practically embedded into DW *ave's* Quantum Annealer given the case that n_q does not surpass the limit of Dwave's Qubit count.

5.5 Computational Complexity of cost function generation



It is worth commenting on the computational complexity of generating the QUBO formulation, as it could become an issue for specific problem instances. Computational complexity can be defined as the number of computer operations to make as a function of input variables. For summa *s*, the complexity becomes proportional to the number of elements to be performed by the summation operator. For , the complexity is equal to O(N). In the case of C(x), the most computationally intensive task is the generation of the 4 nested summations. Assuming the worst case, being that $n' = n = i = lN$, the complexity becomes:

$$O(s^{l}N^{3})$$

which means the complexity is polynomial in time. It is worth considering that, although belonging to the P complexity class, certain implementations of the cost function will require the need super computers. For example, consider the alignment of 20 genomes of influenza A viruses, which roughly contain 13,500 base pairs. This would imply roughly 4.92*e* + 13 operations. Certainly a computationally difficult task that belongs to the domain of super computers, but not untraceable either. Therefore, the feasibility of the cost function generation will depend on the specific problem at hand. Another method to reduce execution time is to consider solely the case of local sequence alignment as opposed to multiple sequence alignment, where l_N becomes several times smaller. Finally, it must be verified that the number of qubits required is within the capacity of the provided Quantum Annealer.

> “
>
> Another method to reduce execution time is to consider solely the case of local sequence alignment as opposed to multiple sequence alignment, where lN becomes several times smaller.

6. Results

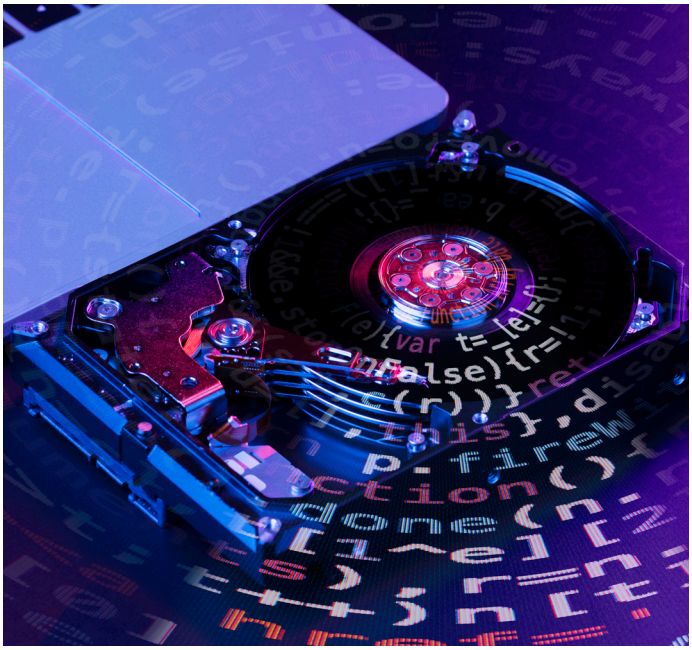
The algorithms were assessed in terms of time to solution and score quality. The Lagrange multipliers where fine-tuned using trial-and-error approaches until no violation of the hard constraints was observed. The values used for the actual tests where 2, 5 and 10 for p_1, p_2 and p3, respectively. Each algorithm was executed 10 times for a fixed number of sequences, and the results were then averaged. This process was repeated for 2 sequences, then 5, and then 5 sequences were subsequently added for every trial, up to 45 sequences, except for the Leap Hybrid Solver which was limited to 40 sequences due to limitations in Leap's qubit count.

6.1 Time to solution

Figures 3 and 4 in appendix A depict the time for algorithm completion as a function of the number of sequences. Given the differences in scale of execution time, separate graphs were made, one displaying compute times for ClustalW and MUSCLE, and the other for Simulated Annealing and the Leap Hybrid Solver, which differ by 3 orders of magnitude. Linear regression was used to describe the tendency of the graphs, obtaining a coefficient of determination no better than 40% for the case of ClustalW and MUSCLE, and 99% for the case of Simulated Annealing and the Leap Hybrid Solver.

6.2 Score results

The aligned sequences where evaluated using the scoring scheme established in 1, where all the column scores are added. For normalization purposes and having the possibility of comparing alignments, the scores where divided by the length of the longest sequence within a group of sequences. This is necessary because MUSCLE and ClustalW can output an alignment that is longer than the length of the longest sequence within a group of sequences to be aligned. Figure 2 from appendix A shows the scores obtained for the different tested algorithms. In terms of score quality, the MUSCLE algorithm showed to preserve the minimum score for all sequence instances. However, in the case of MUSCLE, as the number of sequences is increased, the score value increases, while the scores for Quantum as well as Simulated Annealing seem to stagnate past 10 sequences. ClustalW also seems to preserve the score once past 10 sequences, but the pattern is not as strong as in the case for the Annealing approaches.



7. Conclusions

Owing to this research by NTT DATA,it was possible to set Multiple Sequence Alignment as a QUBO formulation as established in [12], and verified that neither generating the variables or the linear/quadratic interactions require super-polynomial time. Feasible solutions where made possible thanks to the introduction of soft constraints with Lagrange multipliers, and solution feasibility was manually verified. The novelty presented in this paper was testing the QUBO formulation in DWave's hybrid solver, a Quantum Annealer.

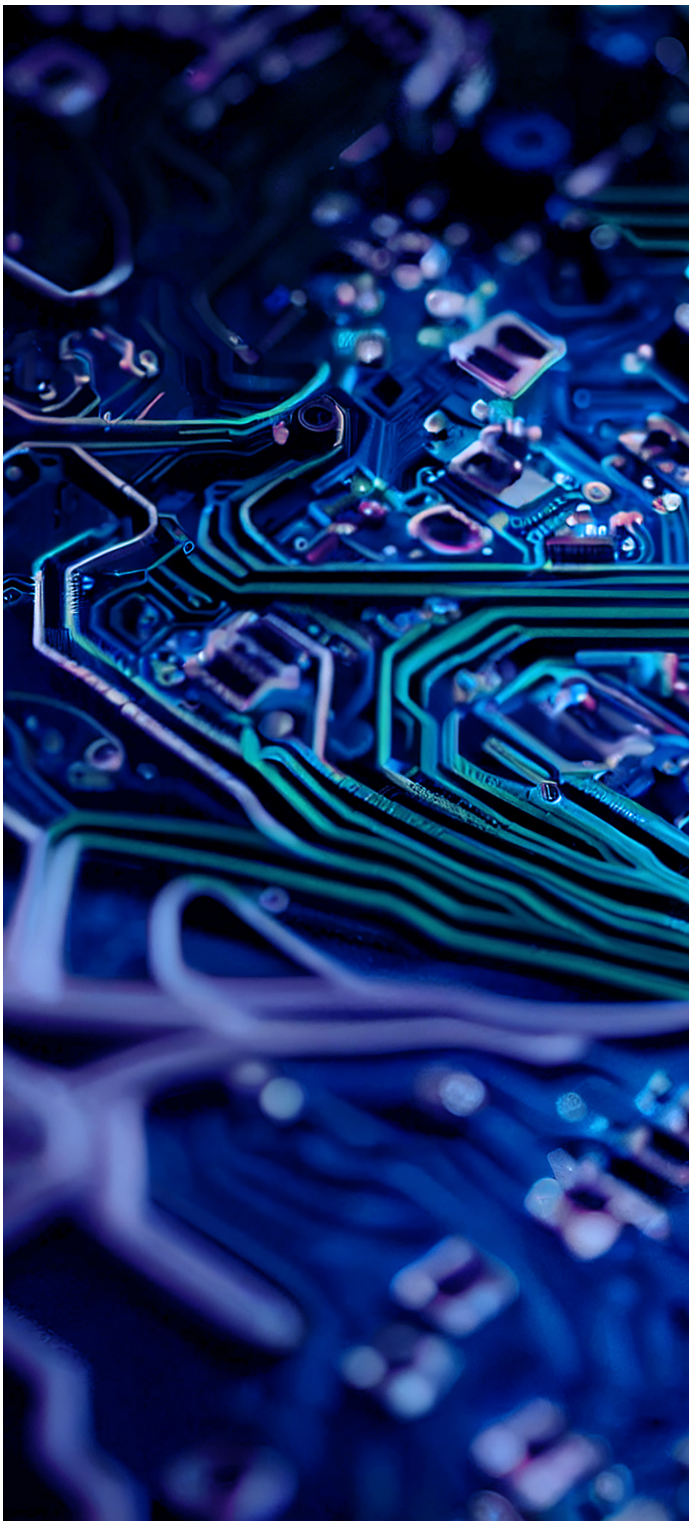
Regarding execution time, ClustalW and MUSCLE solve the problem faster by roughly 3 orders of magnitude than the proposed methodology. On the other hand, it seems to be the case that ClustalW and MUSCLE do not share a mathematical relationship given the low coefficient of determination, but a linear relationship does show up in the case of Simulated Annealing and the Leap Hybrid Solver, with more than 99% accuracy. Moreover, the Hybrid Solver provides a speed-up over Simulated Annealing by some constant factor in this particular case. This could be due to Quantum Annealing's ability to leverage quantum tunneling, allowing the Annealer to cross narrow energy barriers, reaching a solution faster [28]. This is hard to determine given that these energy barriers cannot be visualized due to the Ising model's growing dimensions with the number of qubits.

egarding solution quality, ClustalW and MUSCLE out-performed the Annealing approaches for the set of sequences used, but the scores where close in value when considering more than 30 sequences.

When comparing Quantum vs Simulated Annealing, empirical evidence established in this work suggests that the Quantum Annealing case beats the latter in terms of time to solution for this specific scenario. The speed-up presented by Quantum Annealing over Simulated Annealing should motivate future exploration for the use of Quantum Annealing in cases where Annealing approaches are effective or more suitable that other approaches.

Given the polynomial increase in the number of qubits as a function of the longest sequence length, this approach is more suitable for local alignment rather than global alignment, allowing for a drastic reduction in qubit count.

Finally, when comparing Quantum Annealing with ClustalW and MUSCLE, the Annealing approach underperforms in terms of solution quality and time, but could be the case that it outperforms these algorithms in terms of solution quality if the observed pattern in 6 doesn't break while increasing number of sequences. In order to validate this assumption, Quantum Annealers require a higher qubit count, better qubit fidelity, and longer coherence times; therefore, we are facing an initial contribution with very promising characteristics to boost this crucial field of bioinformatics.



Referencias

[1] W. Cao, LY. Wu, XY. Xia, et al. A sequence-based evolutionary distance method for phylogenetic analysis of highly divergent proteins. *Scientific Reports*, 13:20304, 2023.

[2] Gautam Garai Biswanath Chowdhury. A review on multiple sequence alignment from the perspective of genetic algorithm. *ScienceDirect*, 109:419–431, 2017.

[3] [Authors]. Algorithms for normalized multiple sequence alignments. *ar5iv.org,* 2023.

[4] Ventsislav Kolev, Maria Marinova, and Emilia Pardo. Optimal global sequence alignments of covid-19 nucleotide sequences using the needleman-wunsch algorithm and program parallelization. 2939:020004, 2023.

[5] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology,* 48(3):443–453, 1970.

[6] Christina Beta. Comparison of the needleman-wunsch algorithm and the smith-waterman algorithm for aligning pairs of protein sequences, June 2017. Thesis Examination Committee: Artemis Chatzigeorgiou, Ilias Zintzaras, Chrysoula Doxani.

[7] Fabian Sievers and Desmond G. Higgins. *The Clustal Omega Multiple Alignment Package*, pages 3–16. Springer US, New York, NY, 2021.

[8] R.C. Edgar. Muscle5: High-accuracy alignment ensembles enable unbiased assessments of sequence homology and phylogeny. *Nature Communications,* 13:6968, 2022.

[9] M. Hanussek, F. Bartusch, and J. Krüger. Performance and scaling behavior of bioinformatic applications in virtualization environments to create awareness for the efficient use of compute resources. *PLoS Computational Biology,* 17(7):e1009244, 2021.

[10] B. Kösoglu-Kind, R. Loredo, M. Grossi, et al. A biological sequence comparison algorithm using quantum computers. Scientific Reports, 13:14552, 2023.

[11] A.S. Boev, A.S. Rakitko, S.R. Usmanov, et al. Genome assembly using quantum and quantum-inspired annealing. *Scientific Reports,* 11:13183, 2021.

[12] Sebastian Yde Madsen, Frederik Kofoed Marqversen, Stig Elkjær Rasmussen, and Nikolaj Thomas Zinner. Multi-sequence alignment using the quantum approximate optimization algorithm, 2023.

[13] Various Authors. Applications of next-generation sequencing technologies and computational tools in molecular evolution and aquatic animals conservation studies: A short review. *SAGE* Journals, 2019.

[14] Alex Cabral. The computer science behind dna sequencing, 2019.

[15] Various authors. Technology dictates algorithms: recent developments in read alignment. Genome Biology, 2021.

[16] Jeffrey O'Connell, Amy McNally, and Neil Hall. The real cost of sequencing: scaling computation to keep pace with data generation. *Genome Biology,* 18(1):53, 2017.

[17] Nature Publishing Group. Practical guide for managing large-scale human genome data in research. *Nature*, 2017.

[18] Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[19] David W Mount. *Bioinformatics: Sequence and Genome Analysis.* Cold Spring Harbor Laboratory Press, 2004.

[20] David A. Morrison. Multiple sequence alignment for phylogenetic purposes. *Australian Systematic Botany,* 19(6):479–539, 2006.

[21] Robert C Edgar and Serafim Batzoglou. Multiple sequence alignment. *Current Opinion in Structural Biology,* 16(3):368–373, 2006. Nucleic acids/Sequences and topology.

[22] Julie D Thompson, Toby J Gibson, and Desmond G Higgins. Multiple sequence alignment with the clustal series of programs. *Nucleic acids research,* 31(13):3497–3500, 2003.

[23] Julie D Thompson, Desmond G Higgins, and Toby J Gibson. Improved sensitivity of profile searches through the use of sequence weights and gap excision. Nucleic acids research, 22(22):4673–4680, 1994.

[24] Fabian Sievers and Desmond G Higgins. Clustal omega, accurate alignment of very large numbers of sequences. *Algorithms for Molecular Biology*, 9(1):4, 2014.

[25] Robert C Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004.

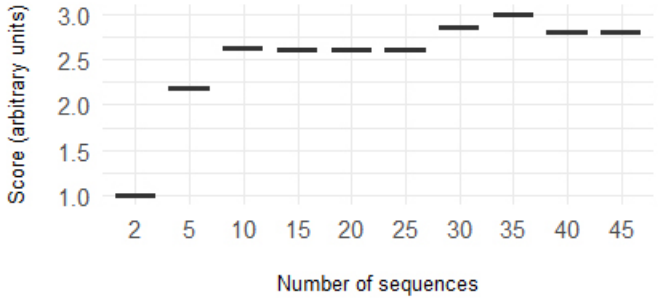
[26] Robert C Edgar. Muscle5: High-accuracy alignment ensembles enable unbiased assessments of sequence homology and phylogeny. Nature Communications, 12(1):1–14, 2021.

[27] Amit Dutta Bikas K. Chakrabarti Atanu Rajak, Sei Suzuki. Quantum annealing: An overview. *The Royal Society,* 2023.

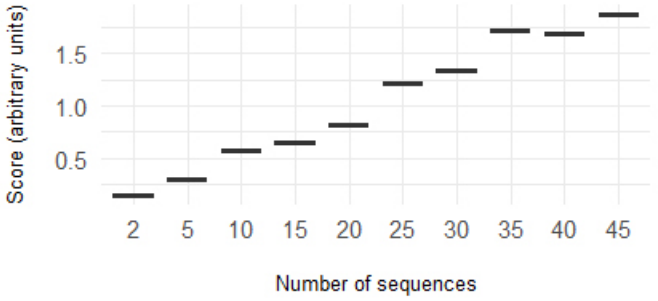
[28] Arnab Das Sei Suzuki. Quantum annealing: The fastest route to quantum computation? *The European Physical Journal,* 2015.

A. Results

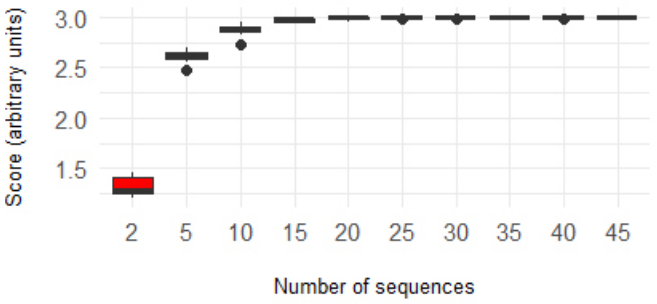
The scores rises when the number of sequence increases for the ClustalW Algorithm



The scores rises when the number of sequence increases for the MUSCLE Algorithm



The scores rises when the number of sequence increases for the simulated Annealing Algorithm



The scores rises when the number of sequence increases for the Leap Hybrid Sampler Algorithm

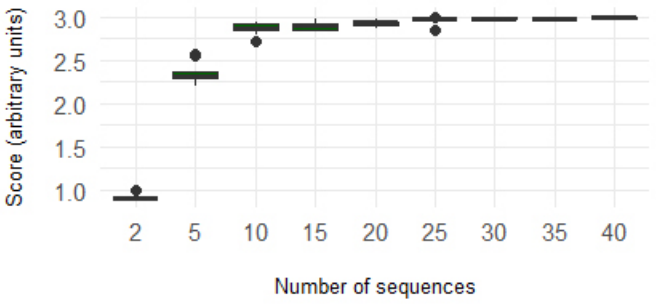


Figure 2: Score as a function of the number of sequences

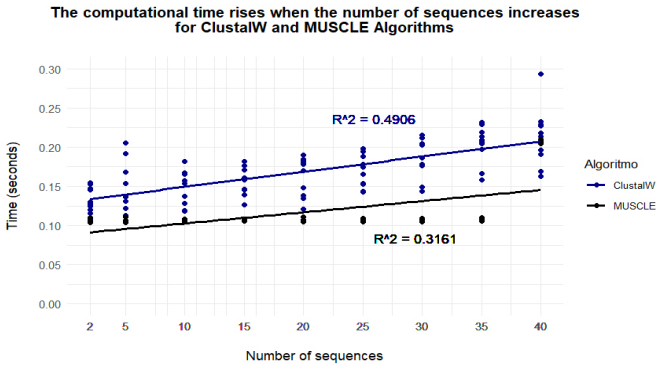


Figure 3: Program execution time as a function of the number of sequences for ClustalW and MUSCLE

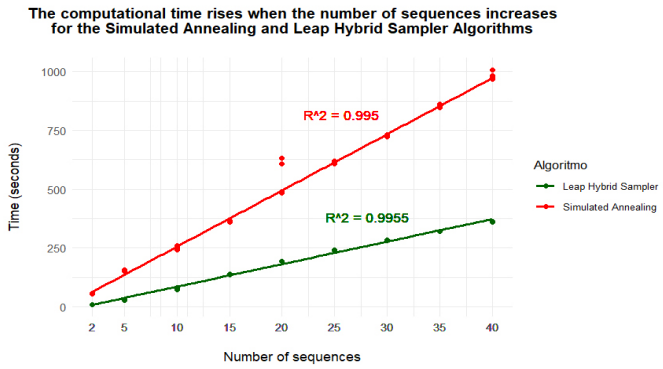


Figure 4: Program execution time as a function of the number of sequences for Simulated Annealing and the Leap Hybrid Solver

pe.nttdata.com