Multi-sequence alignment using the Quantum Approximate Optimization Algorithm

Sebastian Yde Madsen,[1, 2] Frederik Kofoed Marqversen,[1, 2]
Stig Elkjær Rasmussen,[1, 2] and Nikolaj Thomas Zinner[1, 2]

[1]*Department of Physics & Astronomy, Aarhus University, 8000 Aarhus C, Denmark*

[2]*Kvantify Aps, DK-2300 Copenhagen S, Denmark*

The task of *Multiple Sequence Alignment* (MSA) is a constrained combinatorial optimization problem that is generally considered a complex computational problem. In this paper, we first present a binary encoding of MSA and devise a corresponding soft-constrained cost-function that enables a Hamiltonian formulation and implementation of the MSA problem with the variational *Quantum Approximate Optimization Algorithm* (QAOA). Through theoretical analysis, a bound on the ratio of the number of feasible states to the size of the Hilbert space is determined. Furthermore, we consider a small instance of our QAOA-MSA algorithm in both a quantum simulator and its performance on an actual quantum computer. While the ideal solution to the instance of MSA investigated is shown to be the most probable state sampled for a shallow $p < 5$ quantum circuit in the simulation, the level of noise in current devices is still a formidable challenge for the kind of MSA-QAOA algorithm developed here. In turn, we are not able to distinguish the feasible solutions from other states in the quantum hardware output data at this point. This indicates a need for further investigation into both the strategy utilized for compiling the quantum circuit, but also the possibility of devising a more compact ansatz, as one might achieve through constraint-preserving mixers for QAOA.

I. Introduction

A vast variety of present-day technical advances rely heavily on computational models, and their ability to efficiently produce accurate solutions to complex problems. As such, any noticeable improvement in terms of required computational resources, e.g., memory or execution time, for a given problem, can readily be regarded as a very far-reaching and impactful advance to the related scientific fields, and ultimately, everyday technologies.

Inspired by Paul Benioff's early description of a quantum mechanical Turing Machine (TM) [1] and Feynman's 1982 claim that a range of quantum systems can not adequately be simulated on a classical computer [2], David Deutsche, in 1985, claimed that a Universal Turing Machine exists, and that it should be based on a quantum mechanical generalization of the TM, a Quantum TM [3]. This insight was arguably the precursor for the modern field of Quantum Computing, which is in large part concerned with the ability to encode and compute problems, that are classically computationally intractable, through the utilization of quantum mechanical phenomena [4–6].

In the field of bio-informatics, one of the most central and computationally challenging problems is *Multiple Sequence Alignment* (MSA), in which N sequences of protein, DNA, RNA etc. are aligned to one another with regard to some predefined scoring metric, in an attempt to gain valuable insights into, i.a., the functional relationship between said sequences [7–9].

In the present paper, our aim is to try to estimate how a quantum computer may be used to tackle the MSA problem in bio-informatics. We consider both the abstract model of quantum computing, and the question of encoding of MSA into a set of qubits, as well as the more concrete near-term prospects of implementing such a complex calculation on current hardware. The latter effort is directed at variational quantum algorithms, and more precisely the QAOA approach. Simulations are preformed on classical computers to mimic the behavior of a quantum computer in the absence of noise, and these simulations are used to substantiate that QAOA is in principle capable of handling the MSA problem. However, the noise in present quantum hardware could potentially be detrimental to this prospect. By performing a set of very simple tests on small noisy quantum devices, we confirm this expectation. We conclude our presentation with some remarks on the expected hardware improvements in terms of noise-reduction and circuit depths that would be required to start seeing performance on the MSA problem on par with classical computing approaches. A key observation is that improving the QAOA algorithm to restrict it to a subspace of the full Hilbert space would be required in order to improve algorithmic performance, and that this would again imply the need for larger circuit depths at reduced noise levels.

II. Multiple Sequence Alignment

A. Biological outline

From a biological point of view, the DNA molecule is of particular interest as it is the structure ultimately responsible for determining a multitude of functional and observable traits of any organism, i.e., the *Phenotype*. DNA sequencing is the process of determining the nucleotide order of a given DNA fragment, and the ability to sequence effectively is of central importance in biological research. Following the implementation of *Next-Generation Sequencing* technologies [10], the size of available sequence databases has increased many orders of magnitude [11]. Therefore, much of the current research done in fields such as Bioinformatics, is concerned with the development and refinement of principal methods such as *sequence alignment*, which are necessary to perform desirable analyses, such as phylogenetic inference, functional analysis etc. [12, 13] *Pairwise Sequence alignment* (PSA) of DNA, forms the basis of *Multiple Sequence Alignment* (MSA) and can fundamentally be described as the process of mimicking the naturally occurring mutation process of DNA, by inserting gaps in each of the sequences with the intention of improving the vertical agreement of each column [9, 14] (see fig. 1).
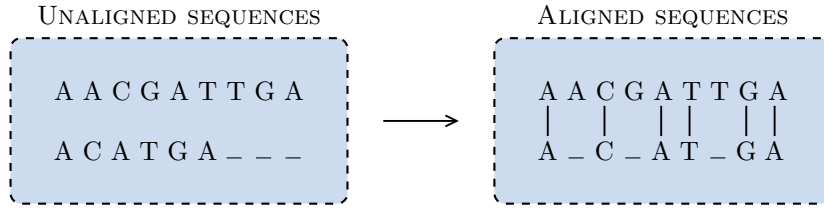

FIG. 1: One possible Pairwise Alignment of 2 sequences.

Multiple Sequence Alignment refers to the process of performing an alignment of three or more sequences. Formally, the process of sequence alignment is subdivided into the two categories of *local alignment* (LA) and *global alignment* (GA), respectively. As suggested by the name, GA is a form of alignment in which the entirety of the sequences are taken into consideration, whereas LA can be described as the process of determining local areas of interest between the sequences and performing alignment based on these. In the following, we will be concerned with the modeling of GA.

B. General problem formulation

Consider a set of $N \geq 2$ strings $S = \{s_i\}_{i=1}^N$, each defined on the alphabet $\Lambda = \{A, C, T, G\}$. Let $l_i = |s_i|$ denote the length of each string, and $L \equiv \max\{l_i\}_{i=1}^N$ the length of the longest string. For $1 \leq j \leq l_i$, let $s_{i,j} \in \Lambda$ denote the jth letter of the ith string, s_i. This setup can be illustrated as follows

$$
\mathbf{S} = \begin{cases} \mathbf{s}_1 &= (s_{1,1} \quad s_{1,2} \quad \ldots \quad s_{1,l_1}) \\ \mathbf{s}_2 &= (s_{2,1} \quad s_{2,2} \quad \ldots \quad s_{2,l_2}) \\ & \quad\quad\quad\quad \vdots \\ \mathbf{s}_N &= (s_{N,1} \quad s_{N,2} \quad \ldots \quad s_{1,l_N}) \end{cases} \quad \text{e.g.} \quad \mathbf{S} = \begin{cases} \mathbf{s}_1 &= (C, \ C, \ T, \ G) \\ \mathbf{s}_2 &= (C, \ G) \\ & \quad \vdots \\ \mathbf{s}_N &= (C, \ T, \ G) \end{cases}
$$

The problem of *Multiple Sequence Alignment* (MSA) is most commonly defined as that of constructing N new strings of equal lengths $l_i = L$ for all i, by inserting into each string an appropriate number of gap-characters '$\_$'.

$$
\mathbf{S} = \begin{pmatrix} s'_{1,1} & s'_{1,2} & \ldots & s'_{1,L} \\ s'_{2,1} & s'_{2,2} & \ldots & s'_{2,L} \\ & & \vdots & \\ s'_{N,1} & s'_{N,2} & \ldots & s'_{1,L} \end{pmatrix} \quad \text{e.g.} \quad \mathbf{S} = \begin{pmatrix} C & C & T & G \\ - & C & - & G \\ & & \vdots & \\ C & - & T & G \end{pmatrix} \tag{1}
$$

The only immediate restriction on a given MSA solution is therefore that it has to preserve the order of the characters. The version of MSA described above does not in general exclude the possibility that two of the initial strings both have maximal length. However, we restrict ourselves to the slightly less general case where exactly one string has maximal length. This string can be viewed as the reference string since it is fixed to its initial ordering.

Note that the exact formulation of a specific instance of MSA is highly dependent upon the biological nature of the problem. If the purpose of the MSA is to perform evolutionary reconstruction, e.g., create a *Phylogenetic tree* of the sequences, then the accuracy of the MSA is highly dependent on the aligned areas of the sequences being homologous. On the other hand, if the MSA is supposed to demonstrate structural equality, the aligned areas of the sequences ought to have comparable positioning in their respective 2 or 3D structures [15].

C. Scoring strategies

Currently, a vast multitude of different scoring schemes exist, both in terms of the computation of intermediate scores in the recursively defined matrices in the *Dynamical Programming* (DP) approach, and in terms of scoring some given MSA. Each of these schemes emphasizes different biological features and as such favor different alignments. The *Sum-of-Pairs* (SP) scoring scheme is one of the most common schemes classically implemented, which assigns a scoring based on the degree of vertical similarity between the sequences, post alignment. As such, given an MSA of the form shown in eq. (1), the SP scheme is defined as the sum of all distinct pairwise comparisons in a column, over all columns

$$\mathrm{SP}(\mathbf{S}') = \sum_{i=1}^{L} \sum_{j=1}^{N} \sum_{j'=j+1}^{N} \mathrm{sim}_{\mathrm{SP}}(s_{j,i}, s_{j',i}), \tag{2}$$

with a native similarity score defined as:

$$\mathrm{sim}_{\mathrm{SP}}(s_{j,i}, s_{j',i}) = \begin{cases} -1, & s_{j,i} = s_{j',i} \\ +1, & s_{j,i} \neq s_{j',i} \\ 0, & s_{j,i} = {}'\_' \ \vee \ s_{j',i} = {}'\_' \end{cases} . \tag{3}$$

D. Intractability of MSA & current approaches

By a combinatorial analysis (see appendix A) it becomes apparent that MSA too suffers from the 'curse of dimensionality', meaning that the number of problem configurations intrinsically expands in at least an exponential manner. Even for the restricted subclass of MSA with an initially fixed number of gaps available for insertion in each sequence, the search space $\mathcal{S}$ is intractably large. In particular, aligning a mere 10 query sequences against a target sequence of 50 characters with 7 gaps available for insertion in each, the number of possible alignments becomes comparable with the number of atoms in the known universe:

$$|\mathcal{S}| = \binom{50}{7}^{9} \sim 10^{79}. \tag{4}$$

As the fastest supercomputer is currently able to perform $\sim 10^{18}$ Flop/s [16], utilizing a brute-force type of search algorithm to generate and evaluate all $\sim 10^{79}$ possible MSA's would still require a completely intractable amount of time. This firmly settles the fact that any such approach is out of the scope of any classical algorithm, assuming of course that one is concerned with a guarantee of finding the global minimum of the chosen scoring metric. This observation is further supported by the fact that MSA is known to be an NP-hard problem under various scoring metrics, including SP [17, 18].

1. Classical approaches to MSA

As PSA forms the basis of MSA, several algorithms have been developed in the past decades [15, 19, 20], in the attempt to provide accurate alignments of multiple sequences efficiently, by initially optimizing the process of aligning a pair of sequences. Most notable instances include the classical DP approach of

the *Needleman–Wunsch* (NW) algorithm for global alignment [21] and *Smith-Waterman* (SM) algorithm for local alignment [22]. For PSA of strings with lengths $\{L_1, L_2\}$, both NW and SM is of time and space complexity $\mathcal{O}(L_1 L_2)$[23], which, for the restricted MSA with N fixed-length strings generalizes to $\mathcal{O}(L^N)$[24].

III. Encodings

To solve the combinatorial problem of finding the best MSA on a quantum computer, the problem has to be encoded in a way that can be represented by a quantum state. Typically, a binary encoding is chosen such that any given binary string, $z \in \{0, 1\}^n$, can be represented by a product state in the *Pauli-Z* basis, e.g.: $001100 \cong |001100\rangle$. The task of defining a valid encoding effectively amounts to finding a binary encoding of the problem which is expressive enough [25, 26] that it can represent any state for a given problem size, but on the other hand, utilizes the shortest possible binary string, to avoid redundancy and computational overhead.

A. One-Hot Column encoding

A straightforward way of encoding any N strings of lengths $(l_i)_{i=1}^N$, arranged in a matrix, is by considering if the nth character in the sth string is placed in the ith column of the matrix, such that each letter would require a binary string of length corresponding the width of the matrix. Specifically, the character A in the string $ACGG$ would be encoded as 1000, as it would inhabit the first column of the matrix. A full example of the One-Hot Column encoding is illustrated in fig. 2 from which it can be seen that the final state, encoding the sequence of strings, is given as a concatenation of the individual 'encoding segments'.
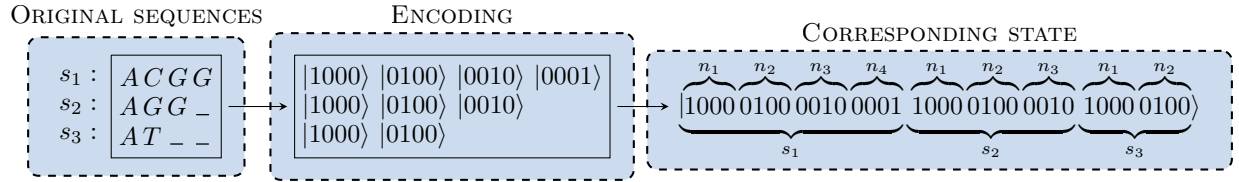
FIG. 2: A generic example showing the *One-Hot column* encoding chosen for this instance of MSA

As the number of columns in the matrix corresponds to the length of the longest string (the *target string*), this encoding generally requires:

$$n = L \sum_{i=1}^N l_i, \quad \text{with} \quad L \equiv \max\{l_i\}_{i=1}^N, \tag{5}$$

qubits to encode, such that the number of qubits scales polynomially in the maximum string length $n = \mathcal{O}(NL^2)$.

B. Cost function

1. Hard constraints

Naturally, almost any encoding enforces a handful of constraints specific to the problem at hand. The One-Hot Column encoding chosen for MSA, entails the following *hard constraints* on feasible solutions:

- Any nth letter in any sth string must occur at exactly one ith of the L available columns:

$$\sum_{i=1}^L x_{s,n,i} = 1., \quad \forall s, n \tag{6}$$

- Any ith of the L available columns, in each row, must be occupied by one letter at most:

$$\sum_{n=1}^{l_s} x_{s,n,i} \leq 1., \quad \forall s, i \tag{7}$$

where l_s denotes the length of the sth string.

Furthermore, any MSA has to preserve the initial order of letters in each of the strings, and as such, can be seen as the most, according to some arbitrary scoring metric, beneficial ordering of gaps between the letters. This formulation of MSA inherently imposes an additional *hard constraint*

- No $(n+1)$th letter in a string must be placed in a column before the nth letter:

$$\sum_{s=1}^{N} x_{s,n',i} x_{s,n,i'} = 0, \quad \forall n < n', \forall i < i'. \tag{8}$$

2. *Scoring scheme*

As the described $x_{s,n,i}$ encoding, alongside the introduced SP-scoring scheme, is incapable of distinguishing different letters of the defined alphabet, Λ, the scheme cannot directly be implemented in the cost function. A possible native way around this issue is an implementation in which all relevant pair-wise scorings of characters in $\mathbf{S}$, are encoded as weights in matrices, as a part of the pre-processing routine. Specifically, for each distinct pair of sequences in a matrix, one would calculate a matrix $[\omega_{i,j}] \equiv \text{score}(s_{s_1,i}, s_{s_2,j})$. As such, preparing $\mathbf{\Omega}$, allows for a scoring equivalent to the SP-scheme to be implemented as:

$$\sum_{s<s'} \sum_{n,n} \sum_{i} \omega_{s,n,s',n'} x_{s,n,i} x_{s',n',i}, \tag{9}$$

in which $\omega_{s,n,s',n'}$ then denotes the predefined scoring associated with alignment of the nth character in the sth string with the (n')th character of the (s')th string.

3. *Soft constraints*

For the resulting cost function to be applicable, the aforementioned *hard constraints* has to be transformed to analogous *soft constraints*, associating each term with some predefined factor, penalizing states in violation of the constraints. In general, this is not a trivial task but Ref. [27] provides a few constraint-penalty pairs, from which the following was inferred:

$$\sum_{i} x_{s,n,i} = 1, \quad \forall s, n \qquad \Longrightarrow \qquad p_1 \sum_{s} \sum_{n} \left(\sum_{i} x_{s,n,i} - 1 \right)^2, \tag{10}$$

$$\sum_{n} x_{s,n,i} \leq 1, \quad \forall s, i \qquad \Longrightarrow \qquad p_2 \sum_{s} \sum_{i} \sum_{n<n'} x_{s,n,i} x_{s,n',i}, \tag{11}$$

$$\sum_{s} x_{s,n',i} x_{s,n,i'} = 0, \quad \forall n < n', \forall i < i' \qquad \Longrightarrow \qquad p_3 \sum_{s} \sum_{n<n'} \sum_{i<i'} x_{s,n',i} x_{s,n,i'}, \tag{12}$$

yielding a final expression for the cost of any binary encoding of some MSA matrix $x \in \{0,1\}^n$ as:

$$\mathcal{C}(x) = \sum_{s<s'} \sum_{n,n'} \sum_{i} \omega_{s,n,s',n'} x_{s,n,i} x_{s',n',i} + p_1 \sum_{s} \sum_{n} \left(\sum_{i} x_{s,n,i} - 1 \right)^2$$
$$+ p_2 \sum_{s} \sum_{i} \sum_{n<n'} x_{s,n,i} x_{s,n',i} + p_3 \sum_{s} \sum_{n<n'} \sum_{i<i'} x_{s,n',i} x_{s,n,i'}. \tag{13}$$

C. Hamiltonian

The fundamental objective is to encode the cost in the Hamiltonian operator, diagonal in the computational *Pauli-Z* basis $\mathcal{C}(x) \longrightarrow \hat{H}(|x\rangle)$ such that the eigenvalues of different feasible MSAs corresponds to eigenvalues

of the Hamiltonian:

$$\hat{\mathcal{H}} |x\rangle = \mathcal{C}(x) |x\rangle, \tag{14}$$

and as the *Pauli-Z* operator has eigenvalues $\{-1, +1\}$

$$\hat{\sigma_z} |y\rangle = (-1)^y |y\rangle, \quad y \in \{0, 1\}. \tag{15}$$

The process of encoding the cost in $\hat{H}$, amounts to the transformation:

$$x_{s,n,i} \to \frac{\mathbb{1} - \hat{\sigma_z}^{s,n,i}}{2}, \tag{16}$$

in which $\hat{\sigma_z}^{s,n,i}$ corresponds to the *Pauli-Z* operator acting on the (s, n, i)th qubit tensor-product state corresponding to the binary encoding of the given MSA matrix. In practice, this is accomplished by initially casting the cost function into a *Quadratic Unconstrained Binary Optimization* (QUBO) model, of the form:

$$C(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{x}^T \mathbf{h} + d, \quad \mathbf{x} \in \{0, 1\}^n, \tag{17}$$

and then mapping the QUBO model to an Ising model. Consequently, this is achieved by decomposing the vectorized QUBO model into a sum and imposing the variable change resulting from eq. (16), which yields:

$$C(\mathbf{s}) = \underbrace{\frac{1}{4}\mathbf{s}^T \mathbf{Q} \mathbf{s}}_{\text{Quadratic term}} - \underbrace{\frac{1}{4}\left(\mathbb{1}_n{}^T \mathbf{Q}^T + \mathbb{1}_n{}^T \mathbf{Q} + 2\mathbf{h}^T\right)\mathbf{s}}_{\text{Linear term}} + \underbrace{\frac{1}{4}\mathbb{1}_n{}^T \mathbf{Q}\mathbb{1}_n + \frac{1}{2}\mathbb{1}_n{}^T \mathbf{h} + d}_{\text{Constant term}},$$

in which $\mathbb{1}_n = \{1\}^n$ is a n-length vector of ones (See appendix B for a more explanatory review of the mathematical transformation).

Specifically, the (i, j)th off-diagonal entries in the quadratic term represent the coupling strength between qubit i and j, and is used as an estimator for the rotation angle in the unitary exponentiation of the two-qubit *Pauli-Z* term in the Hamiltonian acting on qubit i and j, whilst the ith diagonal entry, together with the ith entry in the linear term is used as an estimator of the angle used in the unitary exponentiation of the single-qubit *Pauli-Z* term acting on the ith qubit.

IV.   Variational Quantum Algorithms

Variational Quantum Algorithms (VQAs) are one of the leading strategies within the noisy intermediate-scale quantum (NISQ) era [28, 29]. VQAs seek to provide a very general body for efficiently dealing with computationally complex tasks, which is achieved by means of hybrid quantum-classical (HQC) algorithms, where a classical computer is used to variationally update the variables $\boldsymbol{\theta}$ of a parameterized quantum circuit (PQC), thus attempting to improve upon a purely classical computation, by delegating specific sub-processes to a quantum computer [30]. The underlying principle of VQAs is somewhat inspired by the 'classical' *Variational Method* of quantum mechanics that approximates the ground state energy, E_0, of some system by initially defining some trial state (oftentimes termed *ansatz*) as a function of some parameters, and then minimizes the expectation of the Hamiltonian with respect to these parameters, with the intend of approving upon the approximation of E_0 [31].

A.   Working principle

In the type of VQA considered here, the problem-specific cost function $\mathcal{C}(\boldsymbol{\theta})$ is mapped to an equivalent Hamiltonian $\hat{H}$ which is then encoded into the PQC, such that the action of the cost function on any viable set of parameters, corresponds to 'passing' some initial state $|\psi_0\rangle$ through the PQC. Prior to measurement, parts of the circuit might be transformed to the appropriate measurement basis and the estimate of $\langle\hat{H}\rangle(\boldsymbol{\theta})$ is determined by repeated sampling of the circuit, according to suitable statistical requirements of certainty. Finally, the classical optimizer navigates the cost landscape encoded by the Hamiltonian in the PQC, to determine the parameters $\boldsymbol{\theta^*}$ corresponding to the minimum of the problem-specific cost function:

$$\boldsymbol{\theta^*} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left(\mathcal{C}(\boldsymbol{\theta})\right). \tag{18}$$

The final state of the system is determined upon successive repetitions of sampling the PQC and performing classical minimization, until some predetermined criteria of accuracy are met. Now, as the PQC ultimately corresponds to some unitary transformation $\hat{U}(\boldsymbol{\theta})$, this final state is given as:

$$|\psi(\boldsymbol{\theta})\rangle = \hat{U}(\boldsymbol{\theta})|\psi_0\rangle. \tag{19}$$

B. Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm (QAOA), initially proposed by ref. [32], is a VQA designed for determining approximate solutions to combinatorial optimization problems, inspired by a trotterized version of the Quantum Annealing Algorithm [33, 34]. Originally, the QAOA ansatz was proposed as a repetitive alternation between a problem-specific parameterized unitary and a parameterized 'mixer' unitary. The problem-specific unitary $\hat{U}_P(\gamma) = \mathrm{e}^{-\mathrm{i}\gamma\hat{H}_P}$ is defined by a Hamiltonian

$$\hat{H}_P = \sum_j^K c_j \hat{P}_j^z, \quad \hat{P}_j^z \in \{\hat{\sigma}_z, \hat{\sigma}_z \otimes \hat{\sigma}_z, ..., \hat{\sigma}_z^{\otimes n}\}, \tag{20}$$

diagonal in the space of feasible solutions for the given problem. The cost corresponding to a solution state is encoded in the eigenvalue $\hat{H}_P|\boldsymbol{z}\rangle = \mathcal{C}(\boldsymbol{z})|\boldsymbol{z}\rangle$ of that state. The parameterized 'mixer' unitary $\hat{U}_M(\beta) = \mathrm{e}^{-\mathrm{i}\beta\hat{H}_M}$, is generated by single qubit bit-flips

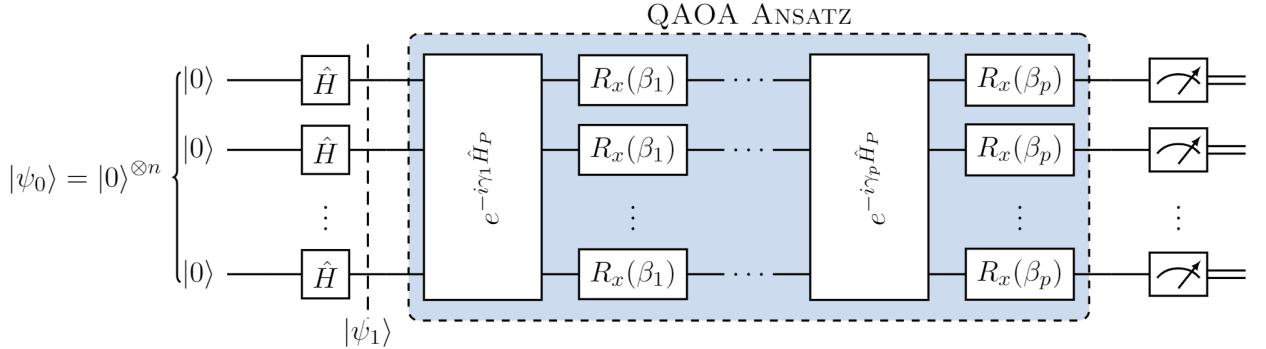$$\mathrm{e}^{-\mathrm{i}\beta\hat{H}_M} = \left(R_x(\beta/2)\right)^{\otimes n}. \tag{21}$$

Preparing the input in the uniform superposition

$$|\psi_1\rangle = \hat{H}^{\otimes n}|0\rangle^{\otimes n} = \left(\frac{1}{\sqrt{2}}|0\rangle + |1\rangle\right)^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle, \tag{22}$$

then produces the following trial state:

$$|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle = \underbrace{\left(\mathrm{e}^{-\mathrm{i}\gamma_1\hat{H}_P}\mathrm{e}^{-\mathrm{i}\beta_1\hat{H}_M}\right)\left(\mathrm{e}^{-\mathrm{i}\gamma_2\hat{H}_P}\mathrm{e}^{-\mathrm{i}\beta_2\hat{H}_M}\right)\cdots\left(\mathrm{e}^{-\mathrm{i}\gamma_p\hat{H}_P}\mathrm{e}^{-\mathrm{i}\beta_p\hat{H}_M}\right)}_{p \text{ repetitions}} |\psi_1\rangle. \tag{23}$$

In conclusion, these steps come together resulting in the circuit shown in fig. 3.

FIG. 3: A generic depiction of the repeating layers of cost and mixer hamiltonians in the QAOA ansatz.

After an appropriate number of shots the VQA strategy would be accompanied by a subroutine of classical optimization, determining:

$$\boldsymbol{\beta}', \boldsymbol{\gamma}' = \underset{\boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathbb{R}^p}{\mathrm{argmin}} \left(\langle\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})|\hat{H}_P|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle\right), \tag{24}$$

and setting:

$$\hat{U}(\boldsymbol{\beta}, \boldsymbol{\gamma}) \rightarrow \hat{U}(\boldsymbol{\beta}', \boldsymbol{\gamma}'), \tag{25}$$

prior to the computation of the next trial state. This effectively means that the circuit depth increases as $\mathcal{O}(p)$ since each repetition of the mixer unitary and Hamiltonian unitary, adds a fixed number of extra gates. From a theoretical point of view, the cardinal advantage of the QAOA stems from the fact that repeated applications of the mixer and the cost unitary yield a high expressivity, such that there exists some $\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*$ for which:

$$\lim_{p \to \infty} \left(\langle \psi(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*) | \hat{H}_P | \psi(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*) \rangle \right) = \mathcal{C}_{\min}, \tag{26}$$

This means that, provided $\boldsymbol{\beta}^*$ and $\boldsymbol{\gamma}^*$ can be efficiently determined, the optimal solution $z^* \in \{0,1\}^n$ can be determined by means of QAOA and corresponds to the bit string encoded in $|\psi(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)\rangle$ [32].

This effectively means that the mixer-terms can be seen as the part of the ansatz assuring that the algorithm is capable of searching the entire Hilbert space, $\mathcal{H}$. However, as the size of $\mathcal{H}$ generally grows exponentially with respect to the number of qubits, n, encoding the problem, i.e., $|\mathcal{H}| = 2^n$, whereas the set of feasible solutions $\mathcal{S}$ to a given problem might only constitute a fraction, i.e., $\mathcal{S} \subseteq \mathcal{H}$. The generic *Pauli-X* mixer chosen in the standard QAOA ansatz suffers from the problem of potentially allowing for the evaluation of many irrelevant states, see figure 4.

It can be seen, that for the specific formulation of MSA and encoding chosen in this project, the ratio, for any $N \geq 2$ strings of general integer lengths $(l_i)_{i=1}^N$, $l_i \in \mathbb{N}$, with a unique maximum length, such that

$$\text{if} \quad L \equiv l_j \equiv \max\{l_i\}_{i=1}^N \quad \text{then} \quad l_j \neq l_i \Leftrightarrow i \neq j, \tag{27}$$

is upper bounded by a function that is super-exponentially decreasing in L and exponentially decreasing in N:

$$\frac{|\mathcal{S}|}{|\mathcal{H}|} \leq \frac{1}{L!} e^{-[\ln(2)L - \ln(L)][L+N-1]}, \quad \forall N, L \in \mathbb{N}_{\geq 2}. \tag{28}$$

See Appendix A for details.
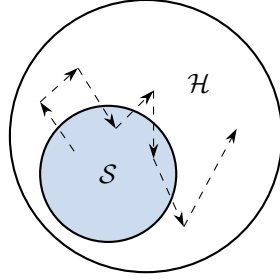
FIG. 4: Venn diagram showing the subspace of feasible solutions, $\mathcal{S}$, being smaller than the Hilbert space $\mathcal{H}$ of all 2^n states, accompanied by a symbolic depiction of the ineffective search of feasible solutions.

This strongly suggests that the encoding chosen for the problem, has a lot of redundancy and that the given combination of encoding and constraints might benefit from specialized constraint-preserving mixers that limit the search to the feasible subspace [35, 36].

1. QAOA Toy Model for MSA

Consider the following native 2-sequence example of MSA:

$$\mathbf{S} = \begin{pmatrix} A\,G \\ G\,- \end{pmatrix} = \begin{pmatrix} |x_{111}x_{112}\rangle & |x_{121}x_{122}\rangle \\ |x_{211}x_{212}\rangle & - \end{pmatrix} \triangleq |x_{111}x_{112}x_{121}x_{122}x_{211}x_{212}\rangle = |100110\rangle, \tag{29}$$

with the previously mentioned weight matrix for the two rows, filled according to the SP scheme:

$$\Omega_{s_1,s_2} = \begin{pmatrix} \omega_{1121} & \omega_{1122} \\ \omega_{1221} & \omega_{1222} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix}. \tag{30}$$

The implementation of the cost function defined in eq. (13), in the QAOA ansatz for a circuit of layer repetition $p = 1$, for the case given in eq. (29) results in the circuit seen on fig. 8 in appendix C.
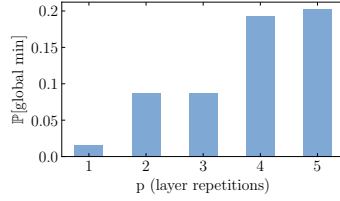
The probability distributions resulting from simulations of this circuit, for an increasing number of repetitions, utilizing Qiskit's framework, alongside the 'classical' Constrained Optimization by Linear Approximation (COBYLA) algorithm [37–39], as implemented by ref. [40], chosen for the parameter optimization subroutine is shown on fig. 5.
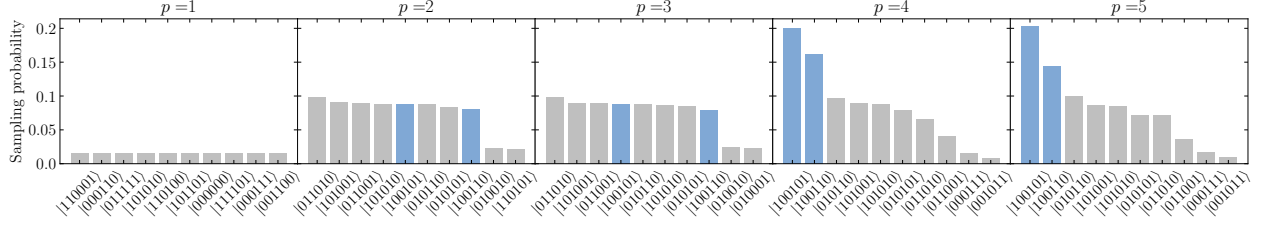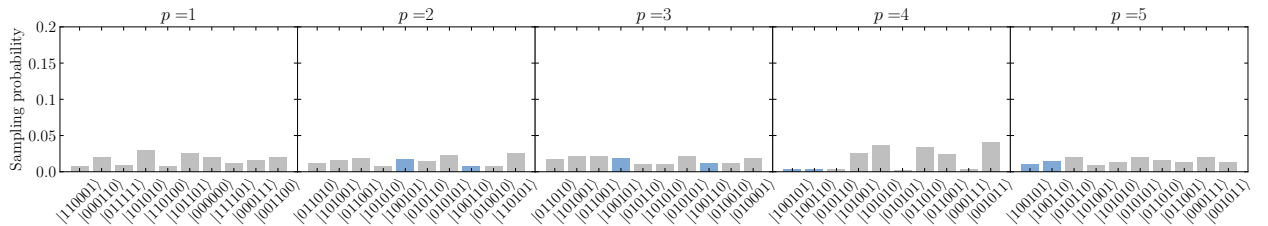

FIG. 5: Histograms depicting the 10 highest probabilities resulting from the simulation of eq. (29), for a QAOA circuit consisting of layer repetitions $p = 1, 2, \ldots, 5$, for 5000 shots. The penalties were set to $p_1 = 10$, $p_2 = p_3 = 1$ in each of the instances. The *blue* bars correspond to feasible solutions.

From the figure, it can be seen that the probability of sampling both the ideal alignment (the global minimum, corresponding to $|100101\rangle$), and, in general, valid alignments, improves, as the number of layer repetitions is increased, and specifically, that the probability of sampling a state corresponding to the global minimum, increases with the number of repetitions, as seen on fig. 6.

FIG. 6: Histogram depicting the probability of sampling the global minimum $|100101\rangle$ as a function of the number of repetitions p, resulting from the simulation of eq. (29), for a QAOA circuit, with 5000 shots taken. The penalties were set to $p_1 = 10$, $p_2 = p_3 = 1$ in each of the instances.

Having these ideal and noiseless results as our baseline, we now proceed to consider how the algorithm performs on actual quantum hardware. In particular, we ran tests on the on Rigetti's Aspen-M-3 79 qubit superconducting quantum computer. The resulting distributions from the quantum hardware are shown in fig. 7.

FIG. 7: Histograms depicting sampling probabilities, resulting from running eq. (29), for a QAOA circuit consisting of layer repetitions $p = 1$, $p = 2, \ldots, p = 5$, for 5000 shots taken, on the Rigetti Aspen-M-3 superconducting quantum computer. The penalties were set to $p_1 = 10$, $p_2 = p_3 = 1$ in each of the instances. It should be noted that the states are ordered, not with respect to sampling probability, but in the same manner as in fig. 6 for comparability. For that reason, the most probable states resulting from these runs, are in general not guaranteed to be visible in the figure. The *blue* bars correspond to feasible solutions.

As evident from the seemingly unordered distributions in fig. 7, the noise level of currently available hardware might still be too significant to enable high probability sampling of feasible solutions for the MSA QAOA type algorithm designed here. Though, it should be noted that the specific choice of circuit-compilation procedure [41], and the chosen subset of qubits for a specific topology influences performance [42], such that better results might very well be achieved through the application of another and more specialized compiler.

V.   Conclusion

By considering Multiple Sequence Alignment as a constrained combinatorial optimization problem, we demonstrated that it is indeed possible to design a binary encoding for the investigated instance of MSA, such that there can be found a 'faithful' cost function which, in turn, can be implemented in a near-term ansatz on a gate based quantum computer.

However, for the proposed near-term QAOA method, a variety of challenges arise. Firstly it is seen that, generally, implementing a cost function with soft constraints, entails additional hyper-parameters in the form of the penalty factors, which in turn has to be optimized, alongside the required number of layer repetitions p, as a part of the pre-processing prior to execution and sampling of the circuit. Furthermore, as near-term quantum devices are prone to noise, any meaningful encoding, and corresponding cost function, has to be designed in a manner that entails sparse requirements to the quantum circuit in terms of gates and qubits required for computation, to avoid faulty computation and the resulting measurement overhead. The specific *one-hot column* encoding chosen for MSA displays heavy redundancy as only an exponentially decreasing fraction of the 2^n different states, representable by the encoding, corresponds to feasible rearrangements of the letters in the MSA matrix, indicating a need for either designing a problem tailored mixer layers in the QAOA ansatz or devising an improved encoding of the problem.

Furthermore, the classical optimization subroutine of the QAOA responsible for estimating the optimal variational parameters, in general, is not uniquely determined, meaning that prior to implementing the problem on a quantum circuit, the choice of routine has to be optimized with respect to problem-specific characteristics. Now, as QAOA was originally intended as a general framework for solving optimization problems on near-term quantum devices, these artifacts pose an immediate challenge with regards to utilizing QAOA for combinatorial optimization.

The QAOA ansatz for the MSA cost-hamiltonian developed here was both simulated using Qiskit's software framework and run on Rigetti's Aspen M-3, both in combination with the classical COBYLA algorithm. While the simulation displays an increasing probability of sampling the feasible states, when increasing the layer repetition parameter p, with the state corresponding to the ideal gap arrangement of the MSA matrix being the most probable for $p \geq 4$, the results from Aspen M-3 exhibited a fairly significant degree of randomness, indicating the presence of an unsuitable amount of noise, and the need for utilizing a more specialized circuit-compiling strategy.

Ultimately, this entails that the ability to efficiently utilize the QAOA ansatz on near-term devices to determine ideal MSAs with respect to the SP-scoring scheme is dependent on gaining further insight into the design of operators capable of restricting the mixing layer to the feasible subspace $\mathcal{S}$, and whether they can be efficiently decomposed and implemented in a gate based device for generally sized instances of MSA, alongside devising a more general strategy for the pre-processing optimization of the hyper-parameters.

Acknowledgments

[1] P. Benioff, The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines, Journal of statistical physics **22**, 563 (1980).

[2] R. P. Feynman, Simulating physics with computers, International Journal of Theoretical Physics **21**, 10.1007/BF02650179 (1982).

[3] D. Deutsch, Quantum theory, the church-turing principle and the universal quantum computer, Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences **400**, 97 (1985).

[4] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, 2022).

[5] S. S. Gill, A. Kumar, H. Singh, M. Singh, K. Kaur, M. Usman, and R. Buyya, Quantum computing: A taxonomy, systematic review and future directions, Software: Practice and Experience **52**, 66 (2022).

[6] J. Preskill, Quantum computing and the entanglement frontier (2012), arXiv:1203.5813 [quant-ph].

[7] P. A. Gagniuc, *Algorithms in bioinformatics: Theory and implementation*, 1st ed. (Wiley, 2021).

[8] D. W. Mount, *Bioinformatics: Sequence and Genome Analysis*, 2nd ed. (Cold Spring Harbor Laboratory Press, 2006).

[9] J. Xiong, *Essential bioinformatics* (Cambridge University Press, 2006).

[10] B. E. Slatko, A. F. Gardner, and F. M. Ausubel, Overview of next generation sequencing technologies, Current protocols in molecular biology **122**, e59 (2018).

[11] K. Sielemann, A. Hafner, and B. Pucker, The reuse of public datasets in the life sciences: Potential risks and rewards, PeerJ **8**, 9 (2020).

[12] D. A. Morrison, Multiple sequence alignment for phylogenetic purposes, Australian Systematic Botany **19**, 479 (2006).

[13] R. C. Edgar and S. Batzoglou, Multiple sequence alignment, Current opinion in structural biology **16**, 368 (2006).

[14] H. Carroll, M. J. Clement, P. Ridge, and Q. O. Snell, Effects of gap open and gap extension penalties, BIOT (2006).

[15] M. Chatzou, C. Magis, J. M. Chang, C. Kemena, G. Bussotti, I. Erb, and C. Notredame, Multiple sequence alignment modeling: methods and applications, Briefings in Bioinformatics **17**, 1009 (2016).

[16] H. W. Meuer, E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer, Top500 - june 2023 (2023), accessed: 2023-07-17.

[17] P. Bonizzoni and G. D. Vedova, The complexity of multiple sequence alignment with sp-score that is a metric, Theoretical Computer Science **259**, 63 (2001).

[18] W. Just, Computational complexity of multiple sequence alignment with sp-score, Journal of computational biology **8**, 615 (2001).

[19] J. Chao, F. Tang, and L. Xu, Developments in algorithms for sequence alignment: A review, Biomolecules 2022, Vol. 12, Page 546 **12**, 546 (2022).

[20] J. D. Thompson, B. Linard, O. Lecompte, and O. Poch, A comprehensive benchmark study of multiple sequence alignment methods: Current challenges and future perspectives, PLoS ONE **6**, 18093 (2011).

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

[22] T. F. Smith and M. S. Waterman, Identification of common molecular subsequences, Journal of Molecular Biology **147**, 195 (1981).

[23] Y. Jararweh, M. Al-Ayyoub, M. Fakirah, L. Alawneh, and B. B. Gupta, Improving the performance of the needleman-wunsch algorithm using parallelization and vectorization techniques, Multimedia Tools and Applications **78**, 3961 (2019).

[24] H. Carrillo and D. Lipman, The multiple sequence alignment problem in biology, SIAM Journal on Applied Mathematics **48**, 1073 (1988).

[25] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms, Advanced Quantum Technologies **2**, 1900070 (2019).

[26] S. E. Rasmussen, N. J. S. Loft, T. Bækkegaard, M. Kues, and N. T. Zinner, Reducing the amount of single-qubit rotations in vqe and related algorithms, Advanced Quantum Technologies **3**, 2000063 (2020).

[27] F. Glover, G. Kochenberger, and Y. Du, A tutorial on formulating and using qubo models (2019), arXiv:1811.11538 [cs.DS].

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

[29] J. W. Z. Lau, K. H. Lim, H. Shrotriya, and L. C. Kwek, Nisq computing: where are we and where do we go?, AAPPS Bulletin **32**, 27 (2022).

[30] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, Nature Reviews Physics **3**, 10.1038/s42254-021-00348-9 (2021).

[31] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, A review on quantum approximate optimization algorithm and its variants (2023), arXiv:2306.09198 [quant-ph].

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

[33] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum computation by adiabatic evolution (2000), arXiv:quant-ph/0001106 [quant-ph].

[34] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem, Science **292**, 472 (2001).

[35] S. Hadfield, Z. Wang, B. Gorman, R. Eleanor, D. Venturelli, and R. Biswas, From the quantum approximate optimization algorithm to a quantum alternating operator ansatz, Algorithms **12**, 10.3390/a12020034 (2019).

[36] F. G. Fuchs, K. O. Lye, H. Nilsen, A. J. Stasik, and G. Sartor, Constraint preserving mixers for the quantum approximate optimization algorithm, Algorithms **15**, 10.3390/a15060202 (2022).

[37] M. J. Powell, *A direct search optimization method that models the objective and constraint functions by linear interpolation* (Springer, 1994).

[38] M. J. Powell, Direct search algorithms for optimization calculations, Acta numerica **7**, 287 (1998).

[39] M. J. Powell, A view of algorithms for optimization without derivatives, Mathematics Today-Bulletin of the Institute of Mathematics and its Applications **43**, 170 (2007).

[40] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods **17**, 261 (2020).

[41] M. Maronese, L. Moro, L. Rocutto, and E. Prati, Quantum compiling, in *Quantum Computing Environments* (Springer, 2022) pp. 39–74.

[42] E. Pelofske, A. Bärtschi, and S. Eidenbenz, Quantum volume in practice: What users can expect from nisq devices, IEEE Transactions on Quantum Engineering **3**, 1 (2022).

A. Complexity analysis of MSA

Number of feasible solutions

Consider a set of N strings $\{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_N\}$ of lengths $\{l_i\}$ with:

$$L = \max\{l_i\}_{i=1}^N. \tag{A1}$$

Say that we want to fill each of the N strings with gaps such that each of them conforms to length L, effectively restricting the number of gaps g_i available for insertion in the $i'th$ string to:

$$g_i = L - l_i. \tag{A2}$$

Because of the fact, that any valid MSA has to preserve the initial ordering of characters, it becomes apparent that the cardinality $|\mathcal{S}|$ of the space $\mathcal{S}$ of feasible solutions, i.e., the number of valid MSAs, for any instance of sequences, can be determined by simple counting, as the problem consequently amounts to counting the number of distinct ways to arrange g_i identical gaps for N different strings.

Particularly, for each string, the first of the g_i identical gaps can be placed in L different spots, the next in $L-1$, and so forth. Hence, the g_i gaps can be arranged in

$$L(L-1)(L-2)\ldots\big(L-(g_i-1)\big) = \frac{L!}{(L-g_i)!} \tag{A3}$$

ways. However, as the gaps are identical objects, symmetry reduces this number to the well-known binomial coefficient:

$$L(L-1)(L-2)\ldots\big(L-(g_i-1)\big)\frac{1}{g_i!} = \frac{L!}{(L-g_i)!g_i!} = \binom{L}{g_i} \tag{A4}$$

As $g_i = 0$ for the string with $L_i = L$, we effectively 'only' have to worry about the arrangement of the $N-1$ strings. However, as:

$$\binom{L}{L} = 1, \quad \forall L \in \mathbb{N}. \tag{A5}$$

The number of feasible solutions can be written as:

$$|\mathcal{S}| = \binom{L}{g_1}\binom{L}{g_2}\cdots\binom{L}{g_N} = \prod_{i=1}^N \binom{L}{g_i}. \tag{A6}$$

Locally upper bounding fraction of feasible states

Following the definition from the previous section, let us establish the fact that the fraction of feasible states:

$$F \equiv \frac{|\mathcal{S}|}{|\mathcal{H}|} = \frac{\prod_{i=1}^N \binom{L}{l_i}}{2^{L\sum_{i=1}^N l_i}}, \tag{A7}$$

is at least exponentially decaying in terms of both the length of the reference string $L = \max_i(l_i)_i$, and the number of strings N. Initially, we define:

$$F_i = \frac{|\mathcal{S}_i|}{|\mathcal{H}_i|} = \frac{\binom{L}{l_i}}{2^{Ll_i}}, \text{ such that } F = \prod_{i=1}^N F_i, \tag{A8}$$

and utilize the general fact that for $1 \leq B \leq A$:

$$\binom{A}{B} \leq \frac{A^B}{B!} \tag{A9}$$

such that:

$$F_i \leq \frac{L^{l_i}}{l_i!} \frac{1}{2^{Ll_i}} = \frac{1}{l_i!} \left(e^{\ln(L)} \right)^{l_i} \left(e^{\ln(2)} \right)^{-Ll_i} = \frac{1}{l_i!} e^{-l_i[\ln(2)L-\ln(L)]}. \tag{A10}$$

Furthermore, in the smallest limiting case, all $l_i = 1$ except the one reference string for which $l_i = L$, and as such:

$$\sum_{i=1}^{N} l_i = L + \sum_{i=1}^{N-1} l_i \geq L + N - 1, \tag{A11}$$

and:

$$\prod_{i=1}^{N} l_i! = L! \prod_{i=1}^{N-1} l_i! \geq L!. \tag{A12}$$

Therefore it is given that:

$$F \leq \prod_{i=1}^{N} \frac{1}{l_i!} e^{-l_i[\ln(2)L-\ln(L)]} \tag{A13}$$

$$\leq \frac{1}{L!} e^{-[\ln(2)L-\ln(L)][L+N-1]} \tag{A14}$$

$$= e^{-\ln(L!)-[\ln(2)L-\ln(L)][L+N-1]}, \quad N, L \in \mathbb{N}_{\geq 2}. \tag{A15}$$

The fraction of feasible states is upper bounded by a function that is super-exponentially decreasing in the length of the reference string L, and exponentially decreasing in the number of strings N.

B. QUBO to Ising

Utilizing the $\{1, 0\} \to \{-1, 1\}$ variable change, yields:

$$C(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} Q_{i,j} x_i x_j + \sum_{i=1}^{n} h_i x_i + d$$

$$\Downarrow$$

$$C(\mathbf{s}) = \sum_{i=1}^{n} \sum_{j=1}^{n} Q_{i,j} \left(\frac{1-s_i}{2} \right) \left(\frac{1-s_j}{2} \right) + \sum_{i=1}^{n} h_i \left(\frac{1-s_i}{2} \right) + d$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{Q_{i,j}}{4} + \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{Q_{i,j}}{4} s_i s_j - \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{Q_{i,j}}{4} s_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{Q_{i,j}}{4} s_j + \sum_{i=1}^{n} \frac{h_i}{2} - \sum_{i=1}^{n} \frac{h_i}{2} s_i + d,$$

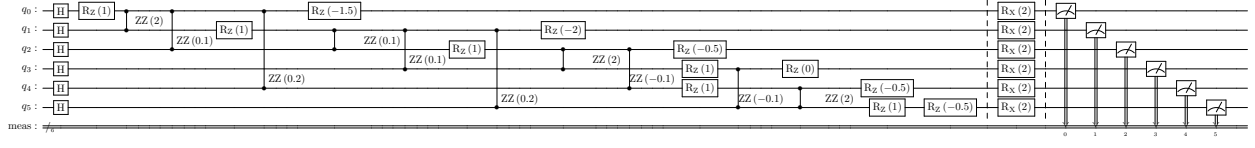which by symmetry yields the familiar grouping:

$$C(\mathbf{s}) = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{Q_{i,j}}{4} s_i s_j \right) - \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{Q_{i,j}+Q_{j,i}}{4} s_i + \sum_{i=1}^{n} \frac{h_i}{2} s_i \right) + \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{Q_{i,j}}{4} + \sum_{i=1}^{n} \frac{h_i}{2} + d \right).$$

Finally, the Ising formulation of (17) is obtained by vectorization, as:

$$C(\mathbf{s}) = \underbrace{\frac{1}{4} \mathbf{s}^T \mathbf{Q} \mathbf{s}}_{\text{Quadratic term}} - \underbrace{\frac{1}{4} \left(\mathbb{1}_n{}^T \mathbf{Q}^T + \mathbb{1}_n{}^T \mathbf{Q} + 2\mathbf{h}^T \right) \mathbf{s}}_{\text{Linear term}} + \underbrace{\frac{1}{4} \mathbb{1}_n{}^T \mathbf{Q} \mathbb{1}_n + \frac{1}{2} \mathbb{1}_n{}^T \mathbf{h} + d}_{\text{Constant term}},$$

where $\mathbb{1}_n$ denotes an $n-$length vector of ones.

C.   Circuit

FIG. 8: $p = 1$ circuit for QAOA ansatz on MSA toy model from eq. (29)