IEEE Transactions on Quantum Engineering

Received 16 March 2023; revised 6 July 2023; accepted 6 July 2023; date of publication 10 July 2023;
date of current version 10 August 2023.

Digital Object Identifier 10.1109/TQE.2023.3293562

Quantum Algorithm for Position Weight Matrix Matching

KOICHI MIYAMOTO[1], NAOKI YAMAMOTO[2,3], AND YASUBUMI SAKAKIBARA[3,4]

[1]Center for Quantum Information and Quantum Biology, Osaka University, Toyonaka 560-0043, Japan
[2]Department of Applied Physics and Physico-Informatics, Keio University, Yokohama 223-8521, Japan
[3]Quantum Computing Center, Keio University, Yokohama 223-8521, Japan
[4]Department of Biosciences and Informatics, Keio University, Yokohama 223-8521, Japan

Corresponding authors: Koichi Miyamoto (e-mail: miyamoto.kouichi.qiqb@osaka-u.ac.jp).

This work was supported by the MEXT Quantum Leap Flagship Program under Grant JPMXS0120319794 and Grant JPMXS0118067285. The work of Koichi Miyamoto was supported by the Japan Society for the Promotion of Science Grants-in-Aid for Scientific Research under Grant JP22K11924, and in part by Japan Science and Technology Agency COI-NEXT program under Grant JPMJPF2014 .

ABSTRACT In this article, we propose two quantum algorithms for a problem in bioinformatics, position weight matrix (PWM) matching, which aims to find segments (sequence motifs) in a biological sequence, such as DNA and protein that have high scores defined by the PWM and are, thus, of informational importance related to biological function. The two proposed algorithms, the naive iteration method and the Monte-Carlo-based method, output matched segments, given the oracular accesses to the entries in the biological sequence and the PWM. The former uses quantum amplitude amplification (QAA) for sequence motif search, resulting in the query complexity scaling on the sequence length n, the sequence motif length m, and the number of the PWMs K as $\widetilde{O}(m\sqrt{Kn})$, which means speedup over existing classical algorithms with respect to n and K. The latter also uses QAA and, further, quantum Monte Carlo integration for segment score calculation, instead of iteratively operating quantum circuits for arithmetic in the naive iteration method; then, it provides the additional speedup with respect to m in some situation. As a drawback, these algorithms use quantum random access memories, and their initialization takes $O(n)$ time. Nevertheless, our algorithms keep the advantage especially when we search matches in a sequence for many PWMs in parallel.

INDEX TERMS Bioinformatics, position weight matrix (PWM) matching, quantum algorithm, quantum amplitude amplification (QAA), quantum Monte Carlo integration (QMCI).

I. INTRODUCTION

Quantum computing [1] is an emerging technology that has a potential to provide large benefits to various fields. Many quantum algorithms that speed up time-consuming problems in classical computing have been proposed, and their applications to practical problems in industry and science have been studied. In this article, we focus on an important problem in bioinformatics: *position weight matrix (PWM) matching*.

As a field in bioinformatics, *sequence analysis*, which focuses on biological sequences such as DNA sequences and protein amino acid sequences, has a long history. Such a sequence is represented as a string of *alphabets* $\mathcal{A}$, e.g., $\{A, G, T, C\}$ for nucleobases in DNA and 20 letters for 20 types of amino acids in a protein, and holds biological information. As a tool to extract important information from a sequence, PWMs, also known as position-specific scoring matrices, are often used. More concretely, a PWM is a tool to find segments with fixed length m that seems to hold specific information from a sequence. A PWM M is a matrix of real values, and its entries reflect the occurrence frequency of each alphabet in a collection of aligned m-length sequences that are similar but different in some positions and thought to be functionally related: for example, if in the ith position, the jth alphabet appears most frequently, the (i, j)th entry of M is set larger than the other entries in the ith row. Conversely, given a PWM M and a sequence, we calculate the score of each m-length segment in the sequence as follows: if the ith position in the segment has the jth alphabet, the (i, j)th entry of M is the score of that position, and the score of the segment is the sum of the scores at all the positions. We then search segments that have scores higher than the predetermined threshold. In this way, we can find some specific patterns (*sequence motifs*) in a sequence, admitting fluctuation of alphabets to some extent. PWMs is in fact used to, for example,

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

find transcription factor binding sites in DNA [2] and infer the 3-D structure of a protein [3].

Following recent developments of next-generation DNA sequencing technology, the volume of data handled in sequence analysis is exponentially growing. Although many classical algorithms and tools for PWM matching have been devised so far [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], it is interesting to investigate the potential of novel technologies, such as quantum computing, to speed up this numerical problem to the extent that classical algorithms cannot reach.

Based on such a motivation, in this article, we propose two quantum algorithms for PWM matching. As far as the authors know, this is the first proposal on quantum algorithms for this problem, although there are some quantum algorithms for exact and/or approximate string match [17], [18], [19], [20].

Our quantum algorithms are twofold: calculating scores of segments and searching high-score segments. For the latter, we use quantum amplitude amplification (QAA) [21], which is a generalization of Grover's algorithm for unstructured database search [22]. As is well known, this provides the quadratic quantum speedup with respect to the number of entries in the database, which now corresponds to n, the length of the sequence.

On the former, we consider two approaches for score calculation, which differentiate the two proposed algorithms. The first one is calculating the segment score by adding the positionwise scores one by one using the quantum circuits for arithmetic. We name the PWM matching quantum algorithm based on this approach the *naive iteration method*. By this method, for any sequence with length n and any K PWMs for sequence motifs with length m, given the oracles to get the specified entry in them, we can find $n_{\rm sol}$ matches with high probability making $\widetilde{O}(m\sqrt{Knn_{\rm sol}})$ queries[1] to the oracles. As far as the authors know, there is no known classical PWM matching algorithm whose worst case complexity is sublinear to n, and thus, the above complexity just shows the quantum speedup. Moreover, note that we aim to search the matches for the multiple PWMs at the same time, which has not been considered in classical algorithms. We achieve the quantum speedup with respect to K too, compared with the K-times sequential runs of the algorithm for the different PWMs, whose complexity obviously scales with K linearly.

The second quantum algorithm uses the quantum Monte Carlo integration (QMCI) [23] to calculate the segment score; we call this method the *QMCI-based method*. QMCI is, similarly to classical Monte Carlo, the method to estimate expectations of random variables, integrals, and sums and also provides the quadratic speedup compared with the classical counterpart. We, therefore, use this to calculate the segment score, which is the sum of the positionwise scores, expecting the further speedup from the naive iteration method, especially when m is large, and thus, the sum has many terms.

[1]The symbol $\widetilde{O}(\cdot)$ hides logarithmic factors in $O(\cdot)$.

This combination of QMCI and QAA is similar to the quantum algorithm for gravitational wave (GW) matched filtering proposed in [24]. A drawback of this approach is the possibility of false detection: the result of QMCI inevitably has an error, and the erroneous estimate on the segment score can exceed the threshold even if the true score does not. To cope with this issue, we introduce two levels of the threshold, $w_{\rm soft}$ and $w_{\rm hard}$, which have the following meaning: we never want to miss segments with scores higher than $w_{\rm hard}$ or falsely find those with scores lower than $w_{\rm soft}$, and it is not necessary but good to find those with scores between $w_{\rm soft}$ and $w_{\rm hard}$. Then, designing the procedure according to evaluation of the error occurring in QMCI calculation of the segment score, by the QMCI-based method, we get all the segments with scores higher than $w_{\rm hard}$ possibly along with some of those with in-between scores with high probability. In this method, we make $\widetilde{O}\left(\frac{mn_{\rm soft}\sqrt{Kn}}{w_{\rm hard}-w_{\rm soft}}\right)$ oracle calls at most, where $n_{\rm soft}$ is the number of segments with scores higher than $w_{\rm soft}$. Although this complexity seemingly has the same dependence on m as the naive iteration method, it can be sublinear to m, since, as explained later, a reasonable choice of $w_{\rm soft}$ and $w_{\rm hard}$ is such that $w_{\rm hard} - w_{\rm soft} \sim \sqrt{m}$.

Although the complexities of the proposed quantum algorithms are sublinear to n and/or m, they require time for preparation. The oracles used in the algorithms can be implemented by quantum random-access memory (QRAM) [25], and the initialization of QRAM, that is, registering the values of the entries in the sequence and the PWMs, takes time, which is estimated as $O(n)$ in usual situations. Despite this initialization cost, our quantum algorithms still have the advantage over the existing classical algorithms, since, among classical ones with $O(n)$ initialization cost, none has the worst case complexity sublinear to n in the main part. Also note that, once we prepare the QRAM for a sequence, our quantum algorithms can search the matches between that sequence and many PWMs, with much smaller initialization cost of the QRAM for the PWMs.

The rest of this article is organized as follows. Section II is preliminary one, where we introduce PWM matching and some building-block quantum algorithms such as QAA and QMCI. Section III is the main part, where we explain our quantum algorithms for PWM matching, the naive iteration method and the QMCI-based method, presenting the detailed procedures in them and the estimations on their complexities. In Section IV, we discuss the aforementioned issues on our algorithms, the preparation cost for the QRAMs, and the plausible setting on the segment score thresholds. Finally, Section V concludes this article.

II. PRELIMINARY

A. NOTATION

We denote the set of all positive real numbers by $\mathbb{R}_+$ and the set of all nonnegative real numbers by $\mathbb{R}_{\geq 0}$.

For each $n \in \mathbb{N}$, we define $[n] := \{1, \dots, n\}$, $[n]_0 := \{0, \dots, n-1\}$, and $\mathbb{N}_{\geq n} := \{m \in \mathbb{N} \mid m \geq n\}$.

For any probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and any random variable X on it, we denote the expectation of X by $E_{\mathbb{P}}[X]$.

For any finite set $\mathcal{A}$ and any $n \in \mathbb{N}$, we denote $S = \{s_0, \ldots, s_{n-1}\} \in \mathcal{A}^n$, where $s_i \in \mathcal{A}$ for each $i \in [n]_0$, by $S = s_0, \ldots, s_{n-1}$.

For any equation or inequality C, $\mathbb{1}_C$ takes 1 if C is satisfied, and 0 otherwise.

For any $x \in \mathbb{R}$, if $|x - y| \leq \epsilon$ holds for some $y \in \mathbb{R}$ and $\epsilon \in \mathbb{R}_+$, we say that x is an ϵ-approximation of y.

B. PWM MATCHING

Here, we formally define the problem we hereafter consider.

Problem 1 (PWM Matching): Suppose that we are given the following:

1) a finite set $\mathcal{A}$ called the alphabet, whose elements are labeled with integers in $[|\mathcal{A}|]_0$;
2) K matrices $M_k = (M_k(i,a))_{i \in [m]_0, a \in \mathcal{A}} \in \mathbb{R}^{m \times |\mathcal{A}|}$, $k \in [K]_0$, called the PWMs, where $K \in \mathbb{N}$ and $m \in \mathbb{N}_{\geq 2}$ is called the *length* of the PWM;
3) an element $S = s_0, \ldots, s_{n-1}$ in $\mathcal{A}^n$, where n is an integer larger than m; we call it a sequence;
4) $w_{\text{th}} \in \mathbb{R}$ called the threshold.

For each $(k, i) \in \mathcal{P}_{\text{all}} := [K]_0 \times [n - m + 1]_0$, define

$$w_{k,i} := W_{M_k}(s_i, \ldots, s_{i+m-1}) \tag{1}$$

where, for each M_k and $u_0, \ldots, u_{m-1} \in \mathcal{A}^m$, we have

$$W_{M_k}(u_0, \ldots, u_{m-1}) := \sum_{j=0}^{m-1} M_k(j, u_j). \tag{2}$$

Then, we want to find all the elements in the set

$$\mathcal{P}_{\text{sol}} := \{(k,i) \in \mathcal{P}_{\text{all}} \mid w_{k,i} \geq w_{\text{th}}\}. \tag{3}$$

Example 1 (PWM Score Calculation): An example of calculating scores for segments using PWM is shown below. The following PWM of length 8 represents the binding site motif for a transcription factor:

A	−1.31	−0.62	−1.31	+0.63	−1.31	−1.31	−1.31	+0.48
C	−0.83	−0.83	+1.12	−0.83	+1.12	+1.12	+1.37	−0.83
G	−0.83	+1.25	+0.27	+0.27	−0.83	+0.27	−0.83	+0.56
T	+0.89	−1.31	−1.31	−1.31	−0.21	−1.31	−1.31	−1.31
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)

Given this PWM M, the score W_M for the DNA sequence (segment) "TACATGCA" is calculated as follows:

$$W_M(\texttt{TACATGCA})$$
$$= \overbrace{+0.89}^{\texttt{T}} \overbrace{-0.62}^{\texttt{A}} \overbrace{+1.12}^{\texttt{C}} \overbrace{+0.63}^{\texttt{A}} \overbrace{-0.21}^{\texttt{T}} \overbrace{+0.27}^{\texttt{G}} \overbrace{+1.37}^{\texttt{C}} \overbrace{+0.48}^{\texttt{A}}$$
$$= 3.93.$$

This PWM matching is then applied to a long genome DNA sequence of million bases such that every segment i in the DNA sequence is assigned a score $W_M(u_i, \ldots, u_{i+m-1})$ and we search $\mathcal{P}_{\text{sol}}$, segments with scores higher than the threshold w_{th}.

In Problem 1, we consider the match with the multiple PWMs simultaneously ($K \geq 2$), although Example 1 is a single-PWM case ($K = 1$). In general, the DNA sequence of a genome contains hundreds of sequence motifs, and the annotation for a genome sequence must be completed by finding all sequence motifs using multiple PWMs simultaneously. As we will see later, we can achieve the quantum speedup with respect to the number K of multiple PWMs, that is, our quantum algorithm finds all the matches between the sequence S and the multiple PWMs $M_0, \ldots, M_{K-1}$ faster than iterating individually searching for the matches between S and each PWM. One might concern that, although it is assumed that all the K PWMs have same length, this is not always the case. This point is easily settled as follows. Denoting the lengths of $M_0, \ldots, M_{K-1}$ by $m_0, \ldots, m_{K-1}$, respectively, we set $m := \max\{m_0, \ldots, m_{K-1}\}$ and, for each $k \in [K]_0$, replace M_k with the $m \times |\mathcal{A}|$ matrix whose first to m_kth rows are those in the original M_k and $(m_k + 1)$th to mth rows are filled with 0. Note that the score of each segment in S does not change under this modification.[2]

The typical orders of magnitudes of the parameters in PWM matching are as follows. The sequence length n can be of order 10^8 (respectively, 10^6–10^7), and the number of PWMs K may be of order 10^2 (respectively, 10^4) for DNA (respectively, protein) [11]. The sequence motif length m is typically about ten or several tens [11], but motifs with lengths greater than 10^2 are sometimes considered for protein [26].

Hereafter, we assume that entries of PWMs are bounded

$$0 \leq M_k(i,a) \leq 1 \tag{4}$$

for every $k \in [K]_0$, $i \in [m]_0$, and $a \in \mathcal{A}$. This is just for the later convenience in using QMCI for score calculation. Although this condition is not satisfied in general cases including Example 1, we can meet it by rescaling. That is, we redefine M'_k as M_k, where

$$M'_k(i,a) := \frac{M_k(i,a) - M_{\min}}{M_{\max} - M_{\min}} \tag{5}$$

with

$$M_{\max} = \max_{(k,i,a) \in [K]_0 \times [m]_0 \times \mathcal{A}} M_k(i,a)$$
$$M_{\min} = \min_{(k,i,a) \in [K]_0 \times [m]_0 \times \mathcal{A}} M_k(i,a). \tag{6}$$

It is easy to see that after this redefinition, (4) holds. We also need to replace the threshold w_{th} with

$$w'_{\text{th}} := \frac{w_{\text{th}} - mM_{\min}}{M_{\max} - M_{\min}}. \tag{7}$$

Note that the set (3) is invariant under the above rescaling.

[2]Note that, for $k \in [K]_0$ such that $m_k < m$, this modification makes the last $m - m_k$ segments with length m_k out of the scope of the matching, although they should be considered. We calculate the scores for such segments and check whether they exceed the threshold, separately from our algorithm. We can reasonably assume that these additional calculations and checks take the negligible time, as far as the number of these exceptional segments, $m - m_k$, is much smaller than that of all the segments, $n - m_k + 1$.

C. QUANTUM ALGORITHMS

Here, we briefly explain the building-block quantum algorithms for our PWM matching algorithm.

1) ARITHMETIC ON A QUANTUM COMPUTER

Before introducing the quantum algorithms, let us summarize the setup for quantum computation and the elementary operations we use in this article.

We consider computation on the system consisting of the multiple quantum registers. We treat real numbers in fixed-point binary representation, and for each $x \in \mathbb{R}$, we denote by $|x\rangle$ the computational basis state on a quantum register where the bit string on the register corresponds to the binary representation of x. We assume that each register has a sufficient number of qubits, and thus, the error from finite-precision representation is negligible.

We use the oracles for elementary arithmetic, such as the adder $O_{\text{add}} |x\rangle |y\rangle = |x\rangle |x+y\rangle$, the subtracter $O_{\text{sub}} |x\rangle |y\rangle = |x-y\rangle |y\rangle$, and the multiplier $O_{\text{mul}} |x\rangle |y\rangle = |x\rangle |xy\rangle$, where $x, y \in \mathbb{R}$. Many proposals on implementations of such oracles have been made so far: see [27] and the references therein.

Besides, we also assume the availability of the following oracles. The oracle $O_=$ checks whether two numbers are equal or not: for any $x, y \in \mathbb{R}$, $O_= |x\rangle |y\rangle |0\rangle = |x\rangle |y\rangle (\mathbb{1}_{x=y} |0\rangle + \mathbb{1}_{x\neq y} |1\rangle)$. Also, the comparator O_{comp} acts as $O_{\text{comp}} |x\rangle |y\rangle |0\rangle = |x\rangle |y\rangle (\mathbb{1}_{x\geq y} |1\rangle + \mathbb{1}_{x<y} |0\rangle)$ for any $x, y \in \mathbb{R}$. These oracles can be implemented via subtraction. To check $x = y$ or not, we may calculate $x - y$ and see whether it is 0 or not. Therefore, we can implement $O_=$ by using a subtracter, followed by a multiple controlled-NOT (CNOT) gate activated if and only if all the bits of $x - y$ are 0, and at last a NOT gate. Moreover, we can implement O_{comp} by combining a subtracter with a CNOT gate activated if and only if the most significant bit of $x - y$ is 0; this is because, if we adopt 2's complement method to represent negative numbers, the most significant bit represents the sign of a number [28].

In addition, for any $N \in \mathbb{N}_{\geq 2}$, we assume the availability of the oracle O_N^{EqPr} that generates the equiprobable superposition of $|0\rangle, |1\rangle, \ldots, |N-1\rangle$: $O_N^{\text{EqPr}} |0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle$. If $N = 2^n$ with some $n \in N$, we can implement this oracle just by operating a Hadamard gate on each qubit of the n-qubit register. If not, letting n be $\lceil \log_2 N \rceil$, we can implement O_N^{EqPr} by the method in [29] to generate a state in which a given probability density function $p(x)$ is amplitude-encoded, with $p(x)$ defined on [0,1] as $p(x) = \mathbb{1}_{x \leq N/2^n}$.

Finally, we use the oracle O_N^{med} that outputs the median $\text{med}(x_1, \ldots, x_N)$ of any N real numbers $x_1, \ldots, x_N$, that is, $O_N^{\text{med}} |x_1\rangle \cdots |x_N\rangle |0\rangle = |x_1\rangle \cdots |x_N\rangle |\text{med}(x_1, \ldots, x_N)\rangle$. The implementations of this oracle have been discussed in [24].

Hereafter, we collectively call the above oracles the arithmetic oracles.

2) QUANTUM AMPLITUDE AMPLIFICATION

The first building block for our PWM matching algorithm is QAA [21]. Given the oracle to generate the superposition state $|\Phi\rangle$, QAA amplifies the amplitude of the "marked state" in $|\Phi\rangle$ so that we can obtain it quadratically faster than naively iterating the process of generating $|\Phi\rangle$ and measurement on it. Here, we give the following theorem, which was presented in [24] as a slight modification of the original one in [21].

Theorem 1 (See [24, Th. 2], Originally [21, Th. 3]): Suppose that we are given an access to an oracle V that acts on the system R consisting of a quantum register R_1 and a qubit R_2, as

$$V |0\rangle |\bar{0}\rangle = \sqrt{a} |\phi_1\rangle |\bar{1}\rangle + \sqrt{1-a} |\phi_0\rangle |\bar{0}\rangle =: |\Phi\rangle \tag{8}$$

where $|\phi_1\rangle$ and $|\phi_0\rangle$ are some quantum states on R_1 and $a \in [0, 1)$. Then, for any $\gamma, \delta \in (0, 1)$, there exists a quantum algorithm $\mathsf{QAA}(V, \gamma, \delta)$ that behaves as follows.

1) The output of the algorithm is either of:
   A) the message "success" and the quantum state $|\phi_1\rangle$;
   B) the message "failure."
2) If $a \geq \gamma$, the algorithm outputs (A) with probability at least $1 - \delta$, making $O(\frac{\log \delta^{-1}}{\sqrt{a}})$ queries to V.
3) If $0 < a < \gamma$, the algorithm outputs either (A) or (B), making $O(\frac{\log \delta^{-1}}{\sqrt{\gamma}})$ queries to V.
4) If $a = 0$, the algorithm certainly outputs (B), making $O(\frac{\log \delta^{-1}}{\sqrt{\gamma}})$ queries to V.[3]

For the detailed procedure of $\mathsf{QAA}(V, \gamma, \delta)$ and the proof of Theorem 1, see [24] and the original paper [21].

3) QUANTUM MONTE CARLO INTEGRATION

QAA leads to quantum amplitude estimation (QAE) algorithm [21], which estimates the amplitude of the marked state; QAE is further extended to the quantum algorithm for estimating the expectation of a random variable [23], which we call QMCI in this article. This is the second building block. There are various versions of QMCI for different situations. For the PWM matching problem, we can use the following one, which assumes that the variable is bounded.

Theorem 2 (See [23, Th. 2.3], Modified): Let $N \in \mathbb{N}$ and $\mathcal{X}$ be a set of N real numbers $X_0, \ldots, X_{N-1}$, each of which satisfies $0 \leq X_i \leq 1$. Suppose that we are given an oracle O_X that acts as

$$O_X |i\rangle |0\rangle = |i\rangle |X_i\rangle \tag{9}$$

for any $i \in [N]_0$. Then, for any ϵ and δ in (0,1), there is an oracle $O_{X,\epsilon,\delta}^{\text{mean}}$ such that

$$O_{X,\epsilon,\delta}^{\text{mean}} |0\rangle = \sum_{y \in \mathcal{Y}} \alpha_y |y\rangle \tag{10}$$

where some ancillary qubits are undisplayed. Here, $\mathcal{Y}$ is a finite set of real numbers that includes a subset $\tilde{\mathcal{Y}}$ consisting

[3] Although [24, Th. 2] does not mention the case that $a = 0$, the statement on this case is proved in Appendix A.

of ϵ-approximations of the mean of $X_0, \ldots, X_{N-1}$

$$\mu := \frac{1}{N}\sum_{i=0}^{N-1} X_i \tag{11}$$

and $\{\alpha_y\}_{y\in\mathcal{Y}}$ are complex numbers satisfying

$$\sum_{\tilde{y}\in\tilde{\mathcal{Y}}} |\alpha_{\tilde{y}}|^2 \geq 1-\delta. \tag{12}$$

In $O^{\text{mean}}_{X,\epsilon,\delta}$,

$$O\left(\frac{1}{\epsilon}\log\left(\frac{1}{\delta}\right)\right) \tag{13}$$

queries to O_X are made.

The proof is presented in Appendix B, where the detailed way to construct the oracle $O^{\text{mean}}_{X,\epsilon,\delta}$ is also shown. Note that this theorem is slightly modified from the original one, [23, Th. 2.3], in some points. First, our Theorem 2 is on the algorithm to calculate the average μ of a sequence and the sum, which is instantly obtained by multiplying the sequence size N to the average. On the other hand, [23, Th. 2.3] presents the algorithm to calculate the expectation of a random variable, and therefore, calculation of the average and the sum is a special case. This is sufficient for us, since we use QMCI to calculate the score of a segment in the sequence S, which is in fact the sum of the scores of the entries in the segment. Second, although the algorithm in [23, Th. 2.3] outputs the approximation of μ, Theorem 2 mentions only generating the state in (10). Although we obtain the approximation of μ by measuring the state, we do not do so in our PWM matching algorithm. This is because our algorithm uses QMCI as a subroutine to calculate the score of each segment in the sequence S in the high-score segment search by QAA. This modification is similar to that in [24], which also presents QMCI with no measurement. However, QMCI in this article is different from that in [24] too, since the former assumes that each X_i is bounded, but the latter assumes that the upper bound on the variance of the sequence is given.

III. QUANTUM ALGORITHM FOR PWM MATCHING

We now present the quantum algorithm for PWM matching. The basic strategy is as follows: we calculate $\{w_{k,i}\}_{k,i}$ for all the pairs $(k,i)\in\mathcal{P}_{\text{all}}$ parallelly in a quantum superposition and find the pairs with high scores by QAA. We present the two versions of the quantum algorithm, the *naive iteration method* and the *QMCI-based method*, whose difference is how to calculate $w_{k,i}$.

A. ASSUMPTION ON THE QUANTUM ACCESSES TO THE SEQUENCE AND THE PWMS

Before presenting the quantum algorithm, we need to make some assumptions on the available oracles. First, for score calculation on a quantum computer, we need to load the entries in the sequence S and the PWMs M_k onto quantum registers. This is formally stated as follows.

Assumption 1: We have accesses to the following oracles.

1) O_{seq}: for any $i\in[n]_0$,

$$O_{\text{seq}}\,|i\rangle\,|0\rangle = |i\rangle\,|s_i\rangle\,. \tag{14}$$

2) O_{PWM}: for any $(k,i,a)\in[K]_0\times[m]_0\times\mathcal{A}$,

$$O_{\text{PWM}}\,|k\rangle\,|i\rangle\,|a\rangle\,|0\rangle = |k\rangle\,|i\rangle\,|a\rangle\,|M_k(i,a)\rangle\,. \tag{15}$$

Here and hereafter, $|a\rangle$ with $a\in\mathcal{A}$ is regarded as the computational basis state corresponding to the integer that labels a. We can implement these oracles if QRAM [25] is available, but nonnegligible preprocessing cost is needed. We will discuss this point in Section IV-A.

Besides, we assume that we can use the oracle that determines whether a given index pair $(k,i)\in\mathcal{P}_{\text{all}}$ is in a given subset $\mathcal{P}\subset\mathcal{P}_{\text{all}}$ or not.

Assumption 2: For any subset $\mathcal{P}\subset\mathcal{P}_{\text{all}}$, we have an access to the oracle $O_{\mathcal{P}}$ that acts as

$$O_{\mathcal{P}}\,|k\rangle\,|i\rangle\,|0\rangle = |k\rangle\,|i\rangle\left(\mathbb{1}_{(k,i)\in\mathcal{P}}\,|0\rangle + \mathbb{1}_{(k,i)\notin\mathcal{P}}\,|1\rangle\right) \tag{16}$$

for any $(k,i)\in\mathcal{P}_{\text{all}}$.

We can also implement this oracle using QRAM, as discussed in Section IV-A.

Since O_{seq}, O_{PWM}, and $O_{\mathcal{P}}$ are supposed to be realized by QRAM, we hereafter consider the number of queries to them as a metric of the complexity of our algorithm.

B. ALGORITHM 1: THE NAIVE ITERATION METHOD

We now explain the first algorithm, the naive iteration method.

We can calculate $w_{k,i}$ by naively iterating the queries to O_{seq} and O_{PWM} and additions as Procedure 1.

Procedure 1: Calculate $w_{k,i}$ by Naive Iteration.

```
Input: $k \in [K]_0, i \in [m]_0$
1 Prepare the quantum registers $R_1, ..., R_7$ and initialize
  $R_1, R_2, R_4$ and the others to $|k\rangle, |i\rangle, |i\rangle$ and $|0\rangle$,
  respectively.
2 for $j = 0, ..., m-1$ do
3     Set $R_5$ to $|s_l\rangle$ by $O_{\text{seq}}$ with the value on $R_4$ being $l$.
4     Set $R_6$ to $|M_k(l,a)\rangle$ by $O_{\text{PWM}}$ with the values on
        $R_1, R_3$ and $R_5$ being $k$, $l$ and $a$, respectively.
5     Using $O_{\text{add}}$, add the value on $R_6$ to the value on $R_7$.
6     if $j < m-1$ then
7         Reset $R_5$ and $R_6$ to $|0\rangle$ using the inverses of
            $O_{\text{seq}}$ and $O_{\text{PWM}}$, respectively.
8         Increment the values on $R_3$ and $R_4$ by 1, using
            $O_{\text{add}}$.
```

In this procedure, the quantum state is transformed as follows:

$$|k\rangle\,|i\rangle\,|0\rangle\,|i\rangle\,|0\rangle\,|0\rangle\,|0\rangle$$

$$\xrightarrow{3}\ |k\rangle\,|i\rangle\,|0\rangle\,|i\rangle\,|s_i\rangle\,|0\rangle\,|0\rangle$$

$$\xrightarrow{4}\ |k\rangle\,|i\rangle\,|0\rangle\,|i\rangle\,|s_i\rangle\,|M_k(0,s_i)\rangle\,|0\rangle$$

$$\xrightarrow{5} |k\rangle\,|i\rangle\,|0\rangle\,|i\rangle\,|s_i\rangle\,|M_k(0,s_i)\rangle\,|M_k(0,s_i)\rangle$$

$$\xrightarrow{7} |k\rangle\,|i\rangle\,|0\rangle\,|i\rangle\,|0\rangle\,|0\rangle\,|M_k(0,s_i)\rangle$$

$$\xrightarrow{8} |k\rangle\,|i\rangle\,|1\rangle\,|i+1\rangle\,|0\rangle\,|0\rangle\,|M_k(0,s_i)\rangle$$

$$\xrightarrow{3} |k\rangle\,|i\rangle\,|1\rangle\,|i+1\rangle\,|s_{i+1}\rangle\,|0\rangle\,|M_k(0,s_i)\rangle$$

$$\xrightarrow{4} |k\rangle\,|i\rangle\,|1\rangle\,|i+1\rangle\,|s_{i+1}\rangle\,|M_k(1,s_{i+1})\rangle\,|M_k(0,s_i)\rangle$$

$$\xrightarrow{5} |k\rangle\,|i\rangle\,|1\rangle\,|i+1\rangle\,|s_{i+1}\rangle\,|M_k(1,s_{i+1})\rangle\left|\sum_{j=0}^{1} M_k(j,s_{i+j})\right\rangle$$

$$\xrightarrow{7} |k\rangle\,|i\rangle\,|1\rangle\,|i+1\rangle\,|0\rangle\,|0\rangle\left|\sum_{j=0}^{1} M_k(j,s_{i+j})\right\rangle$$

$$\to \dots$$

$$\xrightarrow{5} |k\rangle\,|i\rangle\,|m-1\rangle\,|i+m-1\rangle\,|s_{i+m-1}\rangle\,|M_k(m-1,s_{i+m-1})\rangle$$

$$\otimes \left|\underbrace{\sum_{j=0}^{m-1} M_k(j,s_{i+j})}_{=w_{k,i}}\right\rangle. \tag{17}$$

Here, the numbers on the arrows correspond to the steps in Procedure 1. We denote by $O_{\mathrm{sc,it}}$ the quantum circuit for the above operation.

Then, using $O_{\mathrm{sc,it}}$, we can construct the quantum algorithm to find the high-score segments.

Theorem 3: Consider Problem 1 under Assumptions 1 and 2. Suppose that we are given $\delta \in (0,1)$. Then, there exists a quantum algorithm that behaves as follows.

1) If $n_{\mathrm{sol}} := |\mathcal{P}_{\mathrm{sol}}| > 0$, the algorithm outputs all the elements in $\mathcal{P}_{\mathrm{sol}}$ with probability at least $1-\delta$, making

$$O\left(m\sqrt{Knn_{\mathrm{sol}}}\log\left(\frac{Kn}{\delta}\right)\right) \tag{18}$$

queries to O_{seq} and O_{PWM}, and

$$O\left(\sqrt{Knn_{\mathrm{sol}}}\log\left(\frac{Kn}{\delta}\right)\right) \tag{19}$$

queries to $O_{\mathcal{P}}$ with $\mathcal{P}$ being some subsets in $\mathcal{P}_{\mathrm{all}}$.

2) If $\mathcal{P}_{\mathrm{sol}}$ is empty, the algorithm certainly outputs the message "no match," making

$$O\left(m\sqrt{Kn}\log\left(\frac{Kn}{\delta}\right)\right) \tag{20}$$

queries to O_{seq} and O_{PWM}, and

$$O\left(\sqrt{Kn}\log\left(\frac{Kn}{\delta}\right)\right) \tag{21}$$

queries to $O_{\mathcal{P}}$ with $\mathcal{P}$ being some subsets in $\mathcal{P}_{\mathrm{all}}$.

Postponing the proof to Appendix C, we just present the naive iteration method as Algorithm 1. Here, $\tilde{O}^{\mathcal{P}_{\mathrm{temp}}}_{\mathrm{sc,it}}$ is an oracle that generates a quantum state

Algorithm 1: Naive Iteration Method.

```
Input: $\delta \in (0,1)$
1  Set $\mathcal{P}_{\mathrm{temp}} = \emptyset$ and StopFlg $= 0$
2  while StopFlg $= 0$ do
3      Perform QAA$\left(\tilde{O}^{\mathcal{P}_{\mathrm{temp}}}_{\mathrm{sc,it}}, \frac{1}{Kn}, \frac{\delta}{Kn}\right)$.
4      if The output message is "success" then
5          Measure the first and second registers in the
             output state $|\psi_{\mathcal{P}_{\mathrm{temp}},1}\rangle$.
6          Add the measurement outcome $(k,i)$ to $\mathcal{P}_{\mathrm{temp}}$.
7      else
8          Set StopFlg $= 1$
9  if $\mathcal{P}_{\mathrm{temp}} \neq \emptyset$ then
10     Output $\mathcal{P}_{\mathrm{temp}}$.
11 else
12     Output "no match".
```

$$\sqrt{\frac{\left|\mathcal{P}_{\mathrm{sol}} \cap \overline{\mathcal{P}_{\mathrm{temp}}}\right|}{Kn}}\,|\psi_{\mathcal{P}_{\mathrm{temp}},1}\rangle\,|1\rangle + \sqrt{\frac{\left|\overline{\mathcal{P}_{\mathrm{sol}}} \cup \mathcal{P}_{\mathrm{temp}}\right|}{Kn}}\,|\psi_{\mathcal{P}_{\mathrm{temp}},0}\rangle\,|0\rangle \tag{22}$$

where

$$|\psi_{\mathcal{P}_{\mathrm{temp}},1}\rangle := \frac{1}{\sqrt{\left|\mathcal{P}_{\mathrm{sol}} \cap \overline{\mathcal{P}_{\mathrm{temp}}}\right|}}\sum_{(k,i)\in\mathcal{P}_{\mathrm{sol}}\cap\overline{\mathcal{P}_{\mathrm{temp}}}} |k\rangle\,|i\rangle \tag{23}$$

$$|\psi_{\mathcal{P}_{\mathrm{temp}},0}\rangle := \frac{1}{\sqrt{\left|\overline{\mathcal{P}_{\mathrm{sol}}} \cup \mathcal{P}_{\mathrm{temp}}\right|}}\sum_{(k,i)\in\overline{\mathcal{P}_{\mathrm{sol}}}\cup\mathcal{P}_{\mathrm{temp}}} |k\rangle\,|i\rangle \tag{24}$$

and some ancillary registers are omitted in these equations (see Appendix C for details). Here and hereafter, the complement of a set is determined with the universal set being $\mathcal{P}_{\mathrm{all}}$.

Let us comment on the number of qubits used in the naive iteration method. As we see in (17) and (22), this algorithm uses several quantum registers to represent the indexes for PWMs, positions in a sequence, and positions in a segment. The sufficient qubit numbers in these types of registers are $O(\log K)$, $O(\log n)$, and $O(\log m)$, respectively, and thus, the total qubit number is $O(\log K + \log n + \log m)$, logarithmic on the parameters that characterize the problem. Besides, the algorithm uses a few registers to represent real numbers, such as an entry $M_k(j,s)$ of a PWM and a segment score $w_{k,i}$. If we take some typical setting for binary representation of real numbers (say double precision with 64 bits) independently of the problem, a few registers for real numbers amount qubits of order 10^2, which surpasses the qubits for the indexes for typical values of the parameters K, n, and m mentioned in

Section II-B. In summary, for a typical PWM matching problem, the qubit number used in the naive iteration method is of order 10^2. This also applies to the QMCI-based method, which is explained in Section III-C.

C. ALGORITHM 2: QMCI-BASED METHOD

Next, we present the QMCI-based method. It is basically the same as the naive iteration method, but it calculates the score of each segment by QMCI. Although in the naive iteration method, we calculate the score of one segment, which is a sum of the m positionwise scores, calling $O_{\rm seq}$ and $O_{\rm PWM}$ $O(m)$ times, this query number can be reduced by QMCI, whose complexity depends on the required accuracy, if we can set it sufficiently loose.

This method is inspired by the algorithm for GW matched filtering presented in [24]. It is the twofold problem of calculating the quantity called SNR, which is given as the sum of many terms, for each template waveform of the GW signal, and searching the high-SNR templates. In this regard, it has a same structure as PWM matching, which consists of calculating the scores of the segments and searching the high-score segments. Therefore, we naturally conceive the idea to apply the algorithm in [24], which is a combination of QMCI and QAA, to PWM matching.

As pointed out in [24], there is an issue in using QMCI. The result of QMCI inevitably contains the error, and it can cause the *false match*. That is, even if $w_{k,i}$ for some $(k,i) \in \mathcal{P}_{\rm all}$ is smaller than the threshold $w_{\rm th}$, the estimation of it by QMCI might exceed $w_{\rm th}$ due to the error, and we might misjudge that the ith segment matches with M_k. To cope with this, we set the threshold in the similar way to [24]. That is, we set the two levels of the threshold, $w_{\rm soft}$ and $w_{\rm soft}$, which have the following meanings.

1) We want to find $(k,i) \in \mathcal{P}_{\rm all}$ such that $w_{k,i} \geq w_{\rm hard}$ with high probability.
2) We never want to falsely find (k,i) such that $w_{k,i} < w_{\rm soft}$.
3) If there are (k,i) such that $w_{\rm soft} \leq w_{k,i} < w_{\rm hard}$, it is not necessary but fine to find them.

Then, we set the accuracy of QMCI to $\frac{w_{\rm hard}-w_{\rm soft}}{2}$ and take the following policy: if the estimation of $w_{k,i}$ by QMCI is larger than or equal to

$$w_{\rm mid} := \frac{w_{\rm soft}+w_{\rm hard}}{2} \tag{25}$$

we judge (k,i) as "matched," and if not, we judge as "mismatched." Under this policy, (k,i) is judged as "matched" if $w_{k,i} \geq w_{\rm hard}$ and "mismatched" if $w_{k,i} < w_{\rm soft}$ with high probability. We will discuss the validity to assume that such two threshold levels are set in Section IV-B.

Now, let us present the theorem on the QMCI-based method.

Theorem 4: Consider Problem 1 under Assumptions 1 and 2. Suppose that we are given $\delta \in (0,1)$ and $w_{\rm soft}, w_{\rm hard} \in \mathbb{R}$ such that $0 < w_{\rm soft} < w_{\rm hard} < m$. Define

$$\mathcal{P}_{\rm hard} := \{(k,i) \in \mathcal{P}_{\rm all} \mid w_{k,i} \geq w_{\rm hard}\} \tag{26}$$

and

$$\mathcal{P}_{\rm soft} := \{(k,i) \in \mathcal{P}_{\rm all} \mid w_{k,i} \geq w_{\rm soft}\}. \tag{27}$$

Then, there is a quantum algorithm that makes queries to $O_{\rm seq}$, $O_{\rm PWM}$, and $O_{\mathcal{P}}$, with $\mathcal{P}$ being some subsets in $\mathcal{P}_{\rm all}$, and behaves as follows.

1) If $n_{\rm hard} := |\mathcal{P}_{\rm hard}| > 0$, the algorithm outputs all the elements in $\mathcal{P}_{\rm hard}$ and 0 or more elements in $\mathcal{P}_{\rm soft} \setminus \mathcal{P}_{\rm hard}$ with probability at least $1-\delta$. In the algorithm, $O_{\rm seq}$ and $O_{\rm PWM}$ are called
$$O\left(\frac{mn_{\rm soft}\sqrt{Kn}}{w_{\rm hard}-w_{\rm soft}}\log\left(\frac{K^2n^2}{\delta}\right)\log\left(\frac{Kn}{\delta}\right)\right) \tag{28}$$
times, and $O_{\mathcal{P}}$ are called
$$O\left(n_{\rm soft}\sqrt{Kn}\log\left(\frac{K^2n^2}{\delta}\right)\log\left(\frac{Kn}{\delta}\right)\right) \tag{29}$$
times, where $n_{\rm soft} := |\mathcal{P}_{\rm soft}|$.
2) If $n_{\rm soft} = 0$, the algorithm certainly outputs the message "no match." In the algorithm, $O_{\rm seq}$ and $O_{\rm PWM}$ are called
$$O\left(\frac{m\sqrt{Kn}}{w_{\rm hard}-w_{\rm soft}}\log\left(\frac{K^2n^2}{\delta}\right)\log\left(\frac{Kn}{\delta}\right)\right) \tag{30}$$
times, and $O_{\mathcal{P}}$ are called
$$O\left(\sqrt{Kn}\log\left(\frac{K^2n^2}{\delta}\right)\log\left(\frac{Kn}{\delta}\right)\right) \tag{31}$$
times.
3) If $n_{\rm soft} > 0$ and $n_{\rm hard} = 0$, the algorithm certainly outputs the message "no match" or 1 or more elements in $\mathcal{P}_{\rm soft}$. In the algorithm, the number of queries to $O_{\rm seq}$ and $O_{\rm PWM}$ is of order (28), and the number of queries to $O_{\mathcal{P}}$ is of order (29).

Now, we just present the QMCI-based method as Algorithm 2 and leave the proof of Theorem 4 to Appendix D. Here, $\tilde{O}^{\rm sc,QMCI}_{\epsilon,\delta,\mathcal{P}_{\rm temp}}$ is an oracle that by QMCI and some additional operations generates a quantum state

$$\beta_{\mathcal{P}_{\rm temp},1}\left|\xi_{\mathcal{P}_{\rm temp},1}\right\rangle|1\rangle + \beta_{\mathcal{P}_{\rm temp},0}\left|\xi_{\mathcal{P}_{\rm temp},0}\right\rangle|0\rangle\,. \tag{32}$$

$\left|\xi_{\mathcal{P}_{\rm temp},1}\right\rangle$ is a quantum state written as

$$\left|\xi_{\mathcal{P}_{\rm temp},1}\right\rangle := \sum_{(k,i)\in\mathcal{P}_{\rm temp}}\sum_y \alpha^{k,i}_y |k\rangle|i\rangle|y\rangle \tag{33}$$

except the normalization factor, where y runs over a finite set of real numbers larger than $w_{\rm mid}/m$ and the amplitudes $\alpha^{k,i}_y$ concentrate on y's close to $w_{k,i}/m$. $\left|\xi_{\mathcal{P}_{\rm temp},0}\right\rangle$ is another quantum state. $\beta_{\mathcal{P}_{\rm temp},1}$ and $\beta_{\mathcal{P}_{\rm temp},0}$ are complex numbers and $|\beta_{\mathcal{P}_{\rm temp},1}| = \Omega(\sqrt{1/Kn})$. In (32) and (33), some ancillary registers are omitted. For details, see Appendix D.

Algorithm 2: QMCI-Based Method.

```
Input:
  • $\delta \in (0,1)$
  • $w_{\rm soft}$ and $w_{\rm hard}$ such that $0 < w_{\rm soft} < w_{\rm hard} < m$
1  Set $\mathcal{P}_{\rm temp} = \emptyset$, StopFlg $= 0$, $\delta' = \frac{\delta}{4K^2n^2}$, $\delta'' = \frac{\delta}{2Kn}$ and
       $$\epsilon' = \frac{w_{\rm hard} - w_{\rm soft}}{2m}.    (34)$$
2  while StopFlg = 0 do
3      Perform QAA$\left(\tilde{O}^{\rm sc,QMCI}_{\epsilon',\delta',\mathcal{P}_{\rm temp}}, \frac{1}{2Kn}, \delta''\right)$.
4      if The output message is "success" then
5          Measure the first and second registers in the output state $|\xi_{\mathcal{P}_{\rm temp},1}\rangle$ and let the outcome be $(k,i)$.
6          Calculate $w_{k,i}$ classically and let the result be $w^{\rm cl}_{k,i}$.
7          if $w^{\rm cl}_{k,i} \geq w_{\rm soft}$ then
8              Add $(k,i)$ to $\mathcal{P}_{\rm temp}$.
9          else
10             Set StopFlg = 1
11     else
12         Set StopFlg = 1
13 if $\mathcal{P}_{\rm temp} \neq \emptyset$ then
14     Output $\mathcal{P}_{\rm temp}$.
15 else
16     Output "no match".
```

Seemingly, the bounds on the number of queries to $O_{\rm seq}$ and $O_{\rm PWM}$ in (28) and (30) linearly scale with m, which is similar to (18) and (20) and makes us consider that there is no speedup with respect to m compared to the naive iteration method. However, if we can set $w_{\rm soft}$ and $w_{\rm hard}$ with larger difference for larger m, the dependence of the bounds in (28) and (30) on m becomes milder than linear. This seems reasonable because, naively thinking, the typical value of the segment score, which is the sum of m terms, becomes larger for larger m, and so do $w_{\rm soft}$, $w_{\rm hard}$, and their difference. In fact, in Section IV-B, we argue that it is reasonable to take $w_{\rm hard}$ and $w_{\rm soft}$ so that

$$w_{\rm hard} - w_{\rm soft} = \Omega(\sqrt{m}) \tag{35}$$

from which (28) and (30) turn into

$$O\left(n_{\rm soft}\sqrt{Knm}\log\left(\frac{K^2n^2}{\delta}\right)\log\left(\frac{Kn}{\delta}\right)\right) \tag{36}$$

and

$$O\left(\sqrt{Knm}\log\left(\frac{K^2n^2}{\delta}\right)\log\left(\frac{Kn}{\delta}\right)\right) \tag{37}$$

respectively. If so, the QMCI-based method can be beneficial compared to the naive iteration method for small $n_{\rm soft}$ and large m, that is, in the case that there is a small number of matches and the sequence motif length is large.

IV. DISCUSSION

A. IMPLEMENTATIONS OF THE ORACLES WITH QRAMS AND THE COST TO PREPARE THEM

Now, we consider how to implement the oracles $O_{\rm seq}$, $O_{\rm PWM}$, and $O_{\mathcal{P}}$, which we have simply assumed are implementable so far.

It seems that, in order to realize the quantum access to the elements in the sequence S like (14), we need to use a QRAM [25]. Although some difficulties in constructing it in reality have been pointed out [30], it is the very device that provides the access to the indexed data in superposition in $O(\log N)$ time with respect to N the number of the data points. Of course, preparing a QRAM, that is, registering the N data points into the QRAM requires $O(N)$ time. To prepare $O_{\rm seq}$, we need $O(n)$ time.

We can use a QRAM also for $O_{\rm PWM}$ in (15). Although the indexes are now threefold, (k,i,a), it is straightforward to combine them and regard it as an integer. Preparing this takes $O(m|\mathcal{A}|K)$ time, which is expected to be much shorter than $O(N)$ in usual situations.

We can also construct $O_{\mathcal{P}}$, especially $O_{\mathcal{P}_{\rm temp}}$, using a QRAM. Naively thinking, we can do this by registering 0 or 1, which represents $(k,i) \in \mathcal{P}$ or not, for every $(k,i) \in \mathcal{P}_{\rm all}$. However, this takes $O(nK)$ time, which exceeds $O(nm)$ time for the classical exhaustive search if $K > m$. Therefore, we adopt the following approach that takes the shorter time for QRAM preparation. First, we plausibly assume that the number of the matched PWMs at every position i in the sequence S is at most κ, which is $O(1)$. Then, we prepare the QRAM $\tilde{O}_{\mathcal{P}_{\rm temp}}$ that outputs κ indices $k_{i,1},\dots,k_{i,\kappa} \in [K]_0$ such that $(k_{i,1},i),\dots,(k_{i,\kappa},i) \in \mathcal{P}_{\rm temp}$ for each $i \in [n]_0$

$$\tilde{O}_{\mathcal{P}_{\rm temp}}|i\rangle\underbrace{|0\rangle\cdots|0\rangle}_{\kappa} = |i\rangle|k_{i,1}\rangle\cdots|k_{i,\kappa}\rangle. \tag{38}$$

If κ', the number of such indices, is smaller than κ, we set $k_{i,\kappa'+1},\dots,k_{i,\kappa}$ to some dummy number (say, -1) not contained in $[K]_0$. Using this, we can perform the following operation for any $(k,i) \in \mathcal{P}_{\rm all}$:

$$\begin{aligned}
&|k\rangle|i\rangle\underbrace{|0\rangle\cdots|0\rangle}_{\kappa}\underbrace{|0\rangle\cdots|0\rangle}_{\kappa}|0\rangle \\
&\to |k\rangle|i\rangle|k_{i,1}\rangle\cdots|k_{i,\kappa}\rangle|0\rangle\cdots|0\rangle|0\rangle \\
&\to |k\rangle|i\rangle|k_{i,1}\rangle\cdots|k_{i,\kappa}\rangle\left|\mathbb{1}_{k\neq k_{i,1}}\right\rangle \\
&\cdots\left|\mathbb{1}_{k\neq k_{i,\kappa}}\right\rangle|0\rangle \to |k\rangle|i\rangle|k_{i,1}\rangle\cdots|k_{i,\kappa}\rangle\left|\mathbb{1}_{k\neq k_{i,1}}\right\rangle \\
&\cdots\left|\mathbb{1}_{k\neq k_{i,\kappa}}\right\rangle\left|\mathbb{1}_{k\neq k_{i,1}\wedge\cdots\wedge k\neq k_{i,\kappa}}\right\rangle.
\end{aligned} \tag{39}$$

Here, the first to $(\kappa+2)$th kets correspond to registers with the sufficient number of qubits and the other kets correspond to the single qubits. In (39), we use $\tilde{O}_{\mathcal{P}_{\rm temp}}$ at the first arrow and $O_=$'s at the second arrow, and the last operation is done by the multiply controlled NOT gate on the last $\kappa+1$ qubits. Note that "1" on the last qubit means $(k,i) \notin \mathcal{P}_{\rm temp}$. Therefore, the above operation is in fact $O_{\mathcal{P}_{\rm temp}}$, with some registers in (39) regarded as ancillas. In this implementation, the

QRAM $\tilde{O}_{\mathcal{P}_{\rm temp}}$ is queried once in a call to the oracle $O_{\mathcal{P}_{\rm temp}}$, along with $O(\kappa)$ uses of arithmetic oracles. For initializing $\tilde{O}_{\mathcal{P}_{\rm temp}}$, $O(\kappa n)$ time is taken at the very beginning of Algorithm 2, where $\mathcal{P}_{\rm temp} = \emptyset$ and thus $k_{i,1} = \cdots = k_{i,\kappa} = -1$ for any $i \in [n]_0$. After that, every time an index pair (k, i) is added to $\mathcal{P}_{\rm temp}$ in the QAA loop in Algorithm 2, one memory cell in $\tilde{O}_{\mathcal{P}_{\rm temp}}$ is updated, which takes $O(1)$ time.

Let us summarize the above discussion. At the beginning of both the naive iteration method and the QMCI method, we need to initialize the QRAMs $O_{\rm seq}$, $O_{\rm PWM}$, and $\tilde{O}_{\mathcal{P}_{\rm temp}}$, which takes $O(n + m|\mathcal{A}|K + \kappa n)$ time in total. If we reasonably assume that $m|\mathcal{A}|K < n$ and $\kappa = O(1)$, the time complexity is estimated as $O(n)$.

Although we need to take $O(n)$ time at the preliminary stage, after that the quantum algorithms run with complexities shown in Theorems 3 and 4, which scales with n as $O(\sqrt{n})$, for any sequence S and any PWMs M_k. Also note that, once we prepare $O_{\rm seq}$, whose preparation is the bottleneck under the current assumption, we can search the matches between S and another set of K PWMs M'_k by preparing $O_{\rm PWM}$ and $\tilde{O}_{\mathcal{P}_{\rm temp}}$ and running the quantum algorithm, which no longer takes $O(n)$ time.[4] As far as the authors know, there is no known method for PWM matching in which initialization takes $O(n)$ time and the main search algorithm takes the sublinear complexity to n. As an algorithm having the initialization cost of same order, we refer to [11], for example. In this classical algorithm based on an enhanced suffix array (ESA), it takes $O(n)$ time to construct ESA. After that, the worst case complexity to find matches is $O(n + m)$ if some condition is satisfied, but it can be $O(nm)$ in the general case.

B. SCORE THRESHOLD IN THE LARGE m LIMIT

Here, we consider the asymptotic distribution of scores of segment when the sequence motif length m is large and, based on it, discuss the plausible setting on the score threshold.

In many cases, the score threshold $w_{\rm th}$ in matching with a PWM $M \in \mathbb{R}^{m\times|\mathcal{A}|}$ is determined by the p-value. That is, we set $w_{\rm th}$ so that the probability that the score of a segment becomes equal to or larger than $w_{\rm th}$ is equal to the given value $p \in (0, 1)$ in the background model. Here, the background model means the assumption that, when we take a segment of length m in the sequence S randomly and denote by u_i the alphabet in the ith position in the segment, $u_0, \ldots, u_{m-1}$ are independent and identically distributed. The rigorous definition is as follows. Supposing that every $a \in \mathcal{A}$ is associated with $p_a \in (0, 1)$ satisfying $\sum_{a\in\mathcal{A}} p_a = 1$, we consider the finite probability space $(\mathcal{A}^m, \mathbb{P}_{\rm BG})$ consisting of the sample space $\mathcal{A}^m$ and the probability function $\mathbb{P}_{\rm BG} : \mathcal{A}^m \to \mathbb{R}_{\geq 0}$ such that, for any $u_0, \ldots, u_{m-1} \in \mathcal{A}^m$, we have

$$\mathbb{P}_{\rm BG}(u_0, \ldots, u_{m-1}) = \prod_{i=0}^{m-1} p_{a_i} \tag{40}$$

if $u_0 = a_0, \ldots, u_{m-1} = a_{m-1}$ with $a_0, \ldots, a_{m-1} \in \mathcal{A}$. Then, we define

$$w_{\rm th} := \max\{w \in \mathbb{R} | \mathbb{P}_{\rm BG}(\{u_0, \ldots, u_{m-1} \mid W_M(u_0, \ldots, u_{m-1}) \geq w\}) \geq p\} \tag{41}$$

where, for any subset $U \in \mathcal{A}^m$, we define $\mathbb{P}_{\rm BG}(U) := \sum_{u\in U} \mathbb{P}_{\rm BG}(u)$.

Now, we regard $W := W_M(u_0, \ldots, u_{m-1})$ as a random variable and consider its asymptotic distribution in the case of large m. We use the following theorem, a variant of the central limit theorem.

Theorem 5 (See [31, Th. 27.4], Modified): Let $\{X_n\}_{n\in\mathbb{N}_{\geq 0}}$ be the sequence of the independent random variables on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$ such that, for any $n \in \mathbb{N}_{\geq 0}$, X_n has the expectation μ_n and the finite variance σ_n^2. For each $n \in \mathbb{N}$, define $S_n := \sum_{i=0}^{n-1}(X_i - \mu_i)$ and $s_n^2 := \sum_{i=0}^{n-1} \sigma_i^2$. Suppose that there exists $\delta \in \mathbb{R}_+$ such that

$$\lim_{n\to\infty} \frac{1}{s_n^{2+\delta}} \sum_{i=0}^{n-1} E_{\mathbb{P}}[|X_i - \mu_i|^{2+\delta}] = 0. \tag{42}$$

Then, S_n/s_n converges in distribution to a standard normal random variable, as n goes to infinity.

We can apply this theorem to the case of PWM matching by, for each $i \in [m]_0$, regarding X_i in Theorem 5 as $M(i, u_i)$, the score of the alphabet in the ith position in the background model. μ_i and σ_i correspond to the mean and the variance of the score of the alphabet in the ith position, that is,

$$\mu_i = \sum_{a\in\mathcal{A}} p_a M(i, a) \tag{43}$$

and

$$\sigma_i^2 = \sum_{a\in\mathcal{A}} p_a (M(i, a) - \mu_i)^2 \tag{44}$$

respectively. We must check the condition (42) is satisfied, and it is in fact satisfied in the following plausible situation. First, recall that we have rescaled the PWM so that (4) holds. Therefore

$$E_{\mathbb{P}_{\rm BG}}[|X_i - \mu_i|^{2+\delta}] \leq 1 \tag{45}$$

holds obviously. Besides, we may additionally assume that there exist $r \in (0, 1)$ and $\sigma_{\rm min}^2 \in \mathbb{R}_+$ independent of m such that, for at least $\lceil rm \rceil$ elements i in $[m]_0$, $\sigma_i^2 \geq \sigma_{\rm min}^2$ holds. This means that, although in some part of the positions the positionwise score variances might be small, at least in the certain ratio r of the positions, the variances exceeds the level $\sigma_{\rm min}^2$. This assumption yields

$$s_m^2 \geq rm\sigma_{\rm min}^2. \tag{46}$$

[4]Initializing $\tilde{O}_{\mathcal{P}_{\rm temp}}$ seems to take $O(\kappa n)$ time again. However, if we have $\tilde{O}_{\mathcal{P}_{\rm temp}}$ used in the previous algorithm run, resetting its updated memory cells gives us the properly initialized $\tilde{O}_{\mathcal{P}_{\rm temp}}$. Since the number of the memory cells to be reset is equal to that of the matches found in the previous run and it is usually much smaller than n, this reset-based initialization does not take $O(n)$ time.

Combining (45) and (46), we have

$$\frac{1}{s_m^{2+\delta}}\sum_{i=0}^{m-1}E_{\mathbb{P}_{\rm BG}}[|X_i-\mu_i|^{2+\delta}] \le \frac{m}{(rm)^{1+\delta/2}\sigma_{\min}^{2+\delta}} \tag{47}$$

for any $\delta \in \mathbb{R}_+$, which converges to 0 in the large m limit.

Therefore, in the large m limit, we can approximate as

$$\mathbb{P}_{\rm BG}(W \ge w_{\rm th}) \approx \int_{(w_{\rm th}-\tilde{\mu}_m)/s_m}^{\infty} \frac{1}{\sqrt{2\pi}}e^{-x^2/2}dx \tag{48}$$

with $\tilde{\mu}_m := \sum_{i=0}^{m-1}\mu_i = E_{\mathbb{P}_{\rm BG}}[W]$ being the expected segment score under the background model.

Considering the above asymptotic distribution of S, it seems reasonable to set $w_{\rm th}$ as $\tilde{\mu}_m + x\sigma_{\rm tot}$ with some $x \in \mathbb{R}_+$. Alternatively, if we set the two levels of the score $w_{\rm hard}$ and $w_{\rm soft}$ in the QMCI-based method, it seems plausible to set them as $w_{\rm soft} = \tilde{\mu}_m + x_{\rm soft}s_m$ and $w_{\rm hard} = \tilde{\mu}_m + x_{\rm hard}s_m$ with $x_{\rm soft}, x_{\rm hard} \in \mathbb{R}_+$ such that $x_{\rm soft} < x_{\rm hard}$ and $x_{\rm hard} - x_{\rm soft} = O(1)$. For example, $x_{\rm soft} = 3$ and $x_{\rm hard} = 4$, which correspond to the p-values 1.35×10^{-3} and 3.17×10^{-5}, respectively, in the approximation as (48). In such a setting, using $s_m = \Omega(\sqrt{m})$ that follows from (46), we get (35) and then the complexity bounds (36) and (37).

V. CONCLUSION

In this article, we have proposed the two quantum algorithms for an important but time-consuming problem in bioinformatics, PWM matching, which aims to find sequence motifs in a biological sequence whose scores defined by PWMs exceed the threshold. Both of these algorithms, the naive iteration method and the QMCI-based method, utilize QAA for search of high-score segments. They are differentiated by how to calculate the segment score. The former calculates it by simply iterating to add up each positionwise score by quantum circuits for arithmetic. The latter uses QMCI for this summation, coping with false detection due to the QMCI error by setting two levels of threshold, i.e., $w_{\rm soft}$ and $w_{\rm hard}$. Given the oracular accesses to the entries in the sequence and PWMs, both of the quantum algorithms run with query complexity scaling with the sequence length n and the number of PWMs K as $O(\sqrt{Kn})$, thanks to the well-known quadratic speedup by QAA. Furthermore, under some setting on $w_{\rm soft}$ and $w_{\rm hard}$, the complexity of the QMCI-based method scales with the sequence motif length m as $O(\sqrt{m})$. These mean the quantum speedup over existing classical algorithms. Although our quantum algorithms take $O(n)$ preparation time for the initialization of QRAMs, they still have the advantage especially when we perform matching between a sequence and many PWMs.

It is interesting that the QMCI-based method is inspired by the algorithm in [24] for GW astronomy, a completely different field than bioinformatics. From this fact, we expect that the scheme used in the algorithm, the combination of QMCI and QAA, has a large potential for further applications to problems of the "sum-and-search" type over various industrial and scientific fields beyond bioinformatics and astronomy. For example, machine learning might be a promising target, since large-scale matrix multiplication and maximization are common subroutines in widely used models such as convolutional neural network [32] and transformer [33]. In future work, we will explore other applications of quantum algorithms in this type interdisciplinarily.

APPENDIX A
PROOF OF THEOREM 1 FOR THE CASE THAT MAKES $a = 0$

Proof: As described in [24] and the original paper [21], in QAA, we repeatedly generate $G^j|\Phi\rangle$ with various $j \in \mathbb{N}_{\ge 0}$ and measure R_2, and output (A) if and only if the measurement outcome is 1. Here

$$G := -AS_0A^{-1}S_\chi \tag{49}$$

where S_0 and S_χ are the unitary operators on the system under consideration acting as follows:

$$S_\chi|\phi\rangle|x\rangle = \begin{cases} |\phi\rangle|0\rangle, & \text{if } x=0 \\ -|\phi\rangle|1\rangle, & \text{if } x=1 \end{cases} \tag{50}$$

with any state $|\phi\rangle$ on R_1, and

$$S_0|\Phi'\rangle = \begin{cases} -|0\rangle|0\rangle, & \text{if } |\Phi'\rangle = |0\rangle|0\rangle \\ |\Phi'\rangle, & \text{if } \langle\Phi'|(|0\rangle|0\rangle) = 0. \end{cases} \tag{51}$$

As we can see easily, if $a = 0$, $G|\Phi\rangle = |\Phi\rangle = |\phi_0\rangle|0\rangle$ holds, and thus, we never get 1 in measuring R_2 in $G^j|\Phi\rangle$ for any j. Therefore, (A) is never output, which means that (B) is output certainly. ■

For the detailed procedure of $\mathsf{QAA}(A, \gamma, \delta)$ and the proof of Theorem 1 in the other cases, see [24] and the original paper [21].

APPENDIX B
PROOF OF THEOREM 2

Before the proof, we present the following fact, [24, Th. 4]. It is almost same as [23, Lemma 2.1], but slightly modified in reference to the original one, [34, Lemma 6.1].

Lemma 1: Let $\mu \in \mathbb{R}$ and $\epsilon \in \mathbb{R}_+$. Let $\mathcal{A}$ be an algorithm that outputs an ϵ-approximation of μ with probability $\gamma \ge \frac{3}{4}$. Then, for any $\delta \in (0,1)$, the median of outputs in $12\lceil \log\delta^{-1}\rceil + 1$ runs of $\mathcal{A}$ is an ϵ-approximation of μ with probability at least $1-\delta$.

Then, the proof of Theorem 2 is as follows.

Proof of Theorem 2: According to [24, Th. 7], for any integer t larger than 2, we can construct the oracle $\tilde{O}^{\rm mean}_{X,t}$ that acts as $\tilde{O}^{\rm mean}_{X,t}|0\rangle = \sum_{y\in\mathcal{Y}}\alpha_y|y\rangle$ using O_X $O(t)$ times. Here, $\mathcal{Y}$ is a finite set of real numbers that includes a subset $\tilde{\mathcal{Y}}$ consisting of elements $\tilde{\mu}$ satisfying

$$|\tilde{\mu}-\mu| \le C\left(\frac{\sqrt{\mu}}{t}+\frac{1}{t^2}\right) \tag{52}$$

with a universal real constant C, and $\{\alpha_y\}_{y\in\mathcal{Y}}$ are complex numbers satisfying $\sum_{\tilde{y}\in\tilde{\mathcal{Y}}} |\alpha_{\tilde{y}}|^2 \geq 8/\pi^2$. Following this, we prepare a system with J quantum registers and generate the state

$$|\Psi\rangle := \left(\sum_{y_1\in\mathcal{Y}} \alpha_{y_1} |y_1\rangle\right) \otimes \cdots \otimes \left(\sum_{y_J\in\mathcal{Y}} \alpha_{y_J} |y_J\rangle\right) \quad (53)$$

by operating $\tilde{O}^{\text{mean}}_{X,t}$ on each register. Here, J and t are set as

$$J = 12\left\lceil \log \delta^{-1} \right\rceil + 1, \quad t = \left\lceil \frac{2C}{\epsilon} \right\rceil. \quad (54)$$

It can be shown by easy algebra that, in this setting, each $\tilde{\mu} \in \tilde{\mathcal{Y}}$ satisfies $|\tilde{\mu} - \mu| \leq \epsilon$. By measuring $|\Psi\rangle$, we obtain J real numbers $y_1, \ldots, y_J$, each of which is an ϵ-approximation of μ with probability at least $\frac{8}{\pi^2} > \frac{3}{4}$. Therefore, because of Lemma 1, the median of $y_1, \ldots, y_J$ is an ϵ-approximation of μ with probability at least $1 - \delta$. This means that, if we generate

$$|\Psi'\rangle := \left(\sum_{y_1\in\mathcal{Y}} \alpha_{y_1} |y_1\rangle\right) \otimes \cdots \otimes \left(\sum_{y_J\in\mathcal{Y}} \alpha_{y_J} |y_J\rangle\right) |\text{med}(y_1, \ldots, y_J)\rangle \quad (55)$$

by adding one more register to $|\Psi\rangle$ and then using O^{med}_J, this is actually the state in (10), with the first J registers regarded as undisplayed. In summary, we can construct $O^{\text{mean}}_{X,\epsilon,\delta}$ as

$$O^{\text{mean}}_{X,\epsilon,\delta} = O^{\text{med}}_J \left(\underbrace{\tilde{O}^{\text{mean}}_{X,t} \otimes \cdots \otimes \tilde{O}^{\text{mean}}_{X,t}}_{J} \otimes I \right) \quad (56)$$

where I is the identity operator on the Hilbert space for the register. Since each $\tilde{O}^{\text{mean}}_{X,t}$ uses O_X $O(t)$ times, $O^{\text{mean}}_{X,\epsilon,\delta}$ uses O_X $O(tJ)$ times, that is, $O(\frac{1}{\epsilon}\log(\frac{1}{\delta}))$ times, in total. ■

APPENDIX C
PROOF OF THEOREM 3

Proof: First, note that we can perform the following operation for any subset $\mathcal{P} \subset \mathcal{P}_{\text{all}}$:

$$\begin{aligned}
&|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle \\
&\to \frac{1}{\sqrt{Kn}} \sum_{k=0}^{K-1}\sum_{i=0}^{n-1} |k\rangle|i\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle \\
&\to \frac{1}{\sqrt{Kn}} \sum_{k=0}^{K-1}\sum_{i=0}^{n-1} |k\rangle|i\rangle|w_{k,i}\rangle|0\rangle|0\rangle|0\rangle|0\rangle \\
&\to \frac{1}{\sqrt{Kn}} \sum_{k=0}^{K-1}\sum_{i=0}^{n-1} |k\rangle|i\rangle|w_{k,i}\rangle|w_{\text{th}}\rangle|0\rangle|0\rangle|0\rangle \\
&\to \frac{1}{\sqrt{Kn}} \sum_{k=0}^{K-1}\sum_{i=0}^{n-1} |k\rangle|i\rangle|w_{k,i}\rangle|w_{\text{th}}\rangle \\
&\left(\mathbb{1}_{w_{k,i}\geq w_{\text{th}}}|1\rangle + \mathbb{1}_{w_{k,i}<w_{\text{th}}}|0\rangle\right)|0\rangle|0\rangle \\
&\to \frac{1}{\sqrt{Kn}} \sum_{k=0}^{K-1}\sum_{i=0}^{n-1} |k\rangle|i\rangle|w_{k,i}\rangle|w_{\text{th}}\rangle \otimes \\
&\left(\mathbb{1}_{w_{k,i}\geq w_{\text{th}}}|1\rangle + \mathbb{1}_{w_{k,i}<w_{\text{th}}}|0\rangle\right)\left(\mathbb{1}_{(k,i)\in\mathcal{P}}|0\rangle + \mathbb{1}_{(k,i)\notin\mathcal{P}}|1\rangle\right)|0\rangle \\
&\to \frac{1}{\sqrt{Kn}} \sum_{k=0}^{K-1}\sum_{i=0}^{n-1} |k\rangle|i\rangle|w_{k,i}\rangle|w_{\text{th}}\rangle \otimes \\
&\left(\mathbb{1}_{w_{k,i}\geq w_{\text{th}} \wedge (k,i)\notin\mathcal{P}}|1\rangle|1\rangle|1\rangle + \mathbb{1}_{w_{k,i}\geq w_{\text{th}} \wedge (k,i)\in\mathcal{P}}|1\rangle|0\rangle|0\rangle \right.\\
&\left. + \mathbb{1}_{w_{k,i}< w_{\text{th}} \wedge (k,i)\notin\mathcal{P}}|0\rangle|1\rangle|0\rangle + \mathbb{1}_{w_{k,i}< w_{\text{th}} \wedge (k,i)\in\mathcal{P}}|0\rangle|0\rangle|0\rangle\right) \\
&=: \sqrt{\frac{\left|\mathcal{P}_{\text{sol}}\cap\overline{\mathcal{P}}\right|}{Kn}}|\psi_{\mathcal{P},1}\rangle|1\rangle + \sqrt{\frac{\left|\overline{\mathcal{P}_{\text{sol}}\cup\mathcal{P}}\right|}{Kn}}|\psi_{\mathcal{P},0}\rangle|0\rangle \\
&=: |\Psi_{\mathcal{P}}\rangle.
\end{aligned} \quad (57)$$

Here, the seven kets correspond to the seven quantum registers, among which the first four ones have a sufficient number of qubits and the last three ones are single-qubit.

$$|\psi_{\mathcal{P},1}\rangle := \frac{1}{\sqrt{\left|\mathcal{P}_{\text{sol}}\cap\overline{\mathcal{P}}\right|}} \sum_{(k,i)\in\mathcal{P}_{\text{sol}}\cap\overline{\mathcal{P}}} |k\rangle|i\rangle|w_{k,i}\rangle|w_{\text{th}}\rangle|1\rangle|1\rangle \quad (58)$$

is the quantum state on the system consisting of the first to sixth registers, and $|\psi_{\mathcal{P},0}\rangle$ is another state on the same system. In (57), O^{EqPr}_K and O^{EqPr}_n are used at the first arrow. At the second arrow, $O_{\text{sc,it}}$ is used with the register $R_3, \ldots, R_6$ in Procedure 1 undisplayed. At the third arrow, we just set w_{th} on the fourth register. The fourth and fifth transformations are done by O_{comp} and $O_{\mathcal{P}}$, respectively. Then, the last transformation is done by a Toffoli gate on the last three qubits. We denote by $\tilde{O}^{\mathcal{P}}_{\text{sc,it}}$ the oracle for the operation in (57). Note that $\tilde{O}^{\mathcal{P}}_{\text{sc,it}}$ contains one call to $O_{\mathcal{P}}$ and m calls to O_{seq} and O_{PWM}, since $O_{\text{sc,it}}$ makes $O(m)$ calls to them.

Then the naive iteration method is presented as Algorithm 1.

Let us consider the case that $n_{\text{sol}} \geq 1$. In this case, we run QAA repeatedly, and, if each run finishes with the message "success," we obtain an element in $\mathcal{P}_{\text{sol}} \setminus \mathcal{P}_{\text{temp}}$, that is, an element in $\mathcal{P}_{\text{sol}}$ that has not been obtained in the previous runs yet. Thus, if QAAs finish with "success" n_{sol} times in a row, we obtain all the n_{sol} elements in $\mathcal{P}_{\text{sol}}$. This happens with probability at least

$$\left(1 - \frac{\delta}{Kn}\right)^{n_{\text{sol}}} \geq \left(1 - \frac{\delta}{Kn}\right)^{Kn} \geq 1 - \delta \quad (59)$$

since each QAA finishes with the message "success" with probability at least $1 - \frac{\delta}{Kn}$, according to Theorem 1. In the lth QAA, at which the $n_{\text{sol}} - l + 1$ elements in $\mathcal{P}_{\text{sol}}$ remain not to be obtained, the number of the queries to $\tilde{O}^{\mathcal{P}_{\text{temp}}}_{\text{sc,it}}$ is

$$O\left(\sqrt{\frac{Kn}{n_{\text{sol}} - l + 1}} \log\left(\frac{Kn}{\delta}\right)\right) \quad (60)$$

according to Theorem 1, since the amplitude of $|\psi_{\mathcal{P}_{\rm temp},1}\rangle$ in $|\Psi_{\mathcal{P}_{\rm temp}}\rangle$ is

$$\sqrt{\frac{\left|\mathcal{P}_{\rm sol}\cap\overline{\mathcal{P}_{\rm temp}}\right|}{Kn}}=\sqrt{\frac{n_{\rm sol}-l+1}{Kn}}. \tag{61}$$

Thus, in this step, the number of the queries to $O_{\rm seq}$ and $O_{\rm PWM}$ is

$$O\left(m\sqrt{\frac{Kn}{n_{\rm sol}-l+1}}\log\left(\frac{Kn}{\delta}\right)\right) \tag{62}$$

since $\tilde{O}^{\mathcal{P}_{\rm temp}}_{\rm sc,it}$ contains $O(m)$ calls to them, and that of the queries to $O_{\mathcal{P}}$ is of order (60), since $\tilde{O}^{\mathcal{P}_{\rm temp}}_{\rm sc,it}$ calls it once with $\mathcal{P}$ being $\mathcal{P}_{\rm temp}$. Therefore, for $O_{\mathcal{P}}$, the total query number in the series of QAAs is

$$O\left(\sum_{l=1}^{n_{\rm sol}}\sqrt{\frac{Kn}{n_{\rm sol}-l+1}}\log\left(\frac{Kn}{\delta}\right)\right) \tag{63}$$

which turns into (19) by simple algebra, and that for $O_{\rm seq}$ and $O_{\rm PWM}$ is this times m, that is, (18). After $n_{\rm sol}$ QAAs with "success," we run another QAA that outputs "failure" and end the algorithm, since now $\mathcal{P}_{\rm sol}\cap\overline{\mathcal{P}_{\rm temp}}$ is empty and the amplitude of $|\psi_{\mathcal{P}_{\rm temp},1}\rangle$ in $|\Psi_{\mathcal{P}_{\rm temp}}\rangle$ is 0. In this last QAA, the query number for $O_{\rm seq}$ and $O_{\rm PWM}$ is of order (20) and that for $O_{\mathcal{P}}$ is of order (21), according to Theorem 1, but the total query number in the algorithm remains of order (18) and (19).

In the case that $n_{\rm sol}=0$, the first QAA outputs "failure," and then, the algorithm ends. The query number in this is of same order as that in the last QAA in the case that $n_{\rm sol}>0$, that is, (20) and (21). ■

APPENDIX D
PROOF OF THEOREM 4

Proof.

First, note that, for any $k\in[K]_0, i\in[n-m+1]_0$, and $j\in[m]_0$, we can perform the following operation:

$$\begin{aligned}&|k\rangle|i\rangle|j\rangle|0\rangle|0\rangle|0\rangle\\ &\rightarrow|k\rangle|i\rangle|j\rangle|i+j\rangle|0\rangle|0\rangle\\ &\rightarrow|k\rangle|i\rangle|j\rangle|i+j\rangle|s_{i+j}\rangle|0\rangle\\ &\rightarrow|k\rangle|i\rangle|j\rangle|i+j\rangle|s_{i+j}\rangle|M_k(j,s_{i+j})\rangle\end{aligned} \tag{64}$$

where $O_{\rm add}$, $O_{\rm seq}$, and $O_{\rm PWM}$ are used at the first, second, and third arrows, respectively. We denote by $O_{\rm sc,one}$ the oracle for the above operation. According to Theorem 2, for any $\epsilon,\delta\in(0,1)$, we use $O_{\rm sc,one}$ $O(\epsilon^{-1}\log(\delta^{-1}))$ times to construct the oracle $O^{\rm sc,QMCI}_{\epsilon,\delta}$ that acts as

$$O^{\rm sc,QMCI}_{\epsilon,\delta}|k\rangle|i\rangle|0\rangle=|k\rangle|i\rangle\sum_{y\in\mathcal{Y}_{k,i}}\alpha^{k,i}_y|y\rangle. \tag{65}$$

Here, $\mathcal{Y}_{k,i}$ is a finite set of real numbers that includes a subset $\tilde{\mathcal{Y}}_{k,i}$ consisting of ϵ-approximations of $\frac{w_{k,i}}{m}$, and $\{\alpha^{k,i}_y\}_{y\in\mathcal{Y}_{k,i}}$ are complex numbers satisfying

$$\sum_{\tilde{y}\in\tilde{\mathcal{Y}}_{k,i}}|\alpha^{k,i}_{\tilde{y}}|^2\ge1-\delta. \tag{66}$$

Furthermore, with (22), we can construct $\tilde{O}^{\rm sc,QMCI}_{\epsilon,\delta,\mathcal{P}}$ that acts on the seven-register system as

$$\begin{aligned}&\tilde{O}^{\rm sc,QMCI}_{\epsilon,\delta,\mathcal{P}}|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle\\ &=\frac{1}{\sqrt{Kn}}\sum_{k=0}^{K-1}\sum_{i=0}^{n-1}\sum_{y\in\mathcal{Y}_{k,i}}\alpha^{k,i}_y|k\rangle|i\rangle|y\rangle\left|\frac{w_{\rm mid}}{m}\right\rangle\otimes\\ &\Big(\mathbb{1}_{y\ge\frac{w_{\rm mid}}{m}\wedge(k,i)\notin\mathcal{P}}|1\rangle|1\rangle|1\rangle+\mathbb{1}_{y\ge\frac{w_{\rm mid}}{m}\wedge(k,i)\in\mathcal{P}}|1\rangle|0\rangle|0\rangle\\ &+\mathbb{1}_{y<\frac{w_{\rm mid}}{m}\wedge(k,i)\notin\mathcal{P}}|0\rangle|1\rangle|0\rangle+\mathbb{1}_{y<\frac{w_{\rm mid}}{m}\wedge(k,i)\in\mathcal{P}}|0\rangle|0\rangle|0\rangle\Big)\\ &=:\beta_{\mathcal{P},1}|\xi_{\mathcal{P},1}\rangle|1\rangle+\beta_{\mathcal{P},0}|\xi_{\mathcal{P},0}\rangle|0\rangle\\ &=:|\Xi_{\mathcal{P}}\rangle\end{aligned} \tag{67}$$

for any subset $\mathcal{P}\subset\mathcal{P}_{\rm all}$, using $O^{\rm sc,QMCI}_{\epsilon,\delta}$, $O_{\mathcal{P}}$, and some arithmetic oracles. Here

$$\begin{aligned}|\xi_{\mathcal{P},1}\rangle:=&\frac{1}{\sqrt{\sum_{(k,i)\in\overline{\mathcal{P}}}\sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}}|\alpha^{k,i}_y|^2}}\\ &\times\sum_{(k,i)\in\overline{\mathcal{P}}}\sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}}\alpha^{k,i}_y|k\rangle|i\rangle|y\rangle\left|\frac{w_{\rm mid}}{m}\right\rangle|1\rangle|1\rangle\end{aligned} \tag{68}$$

is the quantum states on the first six register, and $|\xi_{\mathcal{P},0}\rangle$ is another state on the same system.

$$\beta_{\mathcal{P},1}=\sqrt{\frac{\sum_{(k,i)\in\overline{\mathcal{P}}}\sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}}|\alpha^{k,i}_y|^2}{Kn}} \tag{69}$$

and $\beta_{\mathcal{P},0}$ is another complex number satisfying $|\beta_{\mathcal{P},0}|^2+|\beta_{\mathcal{P},1}|^2=1$. Note that $\tilde{O}^{\rm sc,QMCI}_{\epsilon,\delta,\mathcal{P}}$ uses $O^{\rm sc,QMCI}_{\epsilon,\delta}$ once, and thus $O_{\rm sc,one}$ $O(\epsilon^{-1}\log(\delta^{-1}))$ times. Since $O_{\rm sc,one}$ calls $O_{\rm seq}$ and $O_{\rm PWM}$ once each, $\tilde{O}^{\rm sc,QMCI}_{\epsilon,\delta,\mathcal{P}}$ calls them $O(\epsilon^{-1}\log(\delta^{-1}))$ times, consequently. Also note that $\tilde{O}^{\rm sc,QMCI}_{\epsilon,\delta,\mathcal{P}}$ uses $O_{\mathcal{P}}$ once.

Then we present the QMCI-based method as Algorithm 2.

Let us consider the behavior of this algorithm in the following cases.

(i) $n_{\rm hard}>0$: For any $(k,i)\in\mathcal{P}_{\rm hard}$

$$\left|y-\frac{w_{k,i}}{m}\right|\le\epsilon'\Rightarrow y\ge\frac{w_{\rm mid}}{m} \tag{70}$$

holds for any $y\in\mathbb{R}$ under the definitions (25) and (34), and thus

$$\sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}}|\alpha^{k,i}_y|^2\ge\sum_{\substack{y\in\mathcal{Y}_{k,i}\\ |y-w_{k,i}|\le\epsilon'}}|\alpha^{k,i}_y|^2\ge1-\delta'\ge\frac{1}{2} \tag{71}$$

holds. This means that, if $\mathcal{P}_{\rm hard} \cap \overline{\mathcal{P}_{\rm temp}} \neq \emptyset$, then

$$\begin{aligned}|\beta_{\mathcal{P}_{\rm temp},1}|^2 &= \frac{\sum_{(k,i)\in\overline{\mathcal{P}}_{\rm temp}} \sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}} |\alpha_y^{k,i}|^2}{Kn} \\ &\ge \frac{\sum_{(k,i)\in\mathcal{P}_{\rm hard}\cap\overline{\mathcal{P}_{\rm temp}}} \sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}} |\alpha_y^{k,i}|^2}{Kn} \\ &\ge \sum_{(k,i)\in\mathcal{P}_{\rm hard}\cap\overline{\mathcal{P}_{\rm temp}}} \frac{1}{2Kn} \\ &\ge \frac{1}{2Kn}\end{aligned} \tag{72}$$

and thus $\mathsf{QAA}\left(\tilde{O}^{\rm sc,QMCI}_{\epsilon',\delta',\mathcal{P}_{\rm temp}}, \frac{1}{2Kn}, \delta''\right)$ outputs $|\xi_{\mathcal{P}_{\rm temp},1}\rangle$ with probability at least $1-\delta'' = 1-\frac{\delta}{2Kn}$.

On the other hand, for $(k,i)\notin\mathcal{P}_{\rm soft}$

$$y \ge \frac{w_{\rm mid}}{m} \Rightarrow \left|y - \frac{w_{k,i}}{m}\right| > \epsilon' \Rightarrow y \notin \tilde{\mathcal{Y}}_{k,i} \tag{73}$$

holds for any $y\in\mathbb{R}$, and thus

$$\sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}} |\alpha_y^{k,i}|^2 = \sum_{\substack{y\in\mathcal{Y}_{k,i}\setminus\tilde{\mathcal{Y}}_{k,i}\\ y\ge w_{\rm mid}/m}} |\alpha_y^{k,i}|^2 \le \sum_{y\in\mathcal{Y}_{k,i}\setminus\tilde{\mathcal{Y}}_{k,i}} |\alpha_y^{k,i}|^2 < \delta' \tag{74}$$

holds. From this, the probability that we obtain $(k,i)\in\mathcal{P}_{\rm soft}$ in measuring the first two registers in $|\xi_{\mathcal{P}_{\rm temp},1}\rangle$ is evaluated as

$$\begin{aligned}&\frac{\sum_{(k,i)\in\mathcal{P}_{\rm soft}\cap\overline{\mathcal{P}_{\rm temp}}} \sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}} |\alpha_y^{k,i}|^2}{\sum_{(k,i)\in\overline{\mathcal{P}_{\rm temp}}} \sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}} |\alpha_y^{k,i}|^2} \\ &= 1 - \frac{\sum_{(k,i)\in\overline{\mathcal{P}_{\rm soft}}\cap\overline{\mathcal{P}_{\rm temp}}} \sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}} |\alpha_y^{k,i}|^2}{\sum_{(k,i)\in\overline{\mathcal{P}_{\rm temp}}} \sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}} |\alpha_y^{k,i}|^2} \\ &\ge 1 - 2\sum_{(k,i)\in\overline{\mathcal{P}_{\rm soft}}\cap\overline{\mathcal{P}_{\rm temp}}} \delta' \\ &\ge 1 - 2Kn\delta' \\ &= 1 - \frac{\delta}{2Kn}.\end{aligned} \tag{75}$$

At the first inequality, we used (74) and

$$\sum_{(k,i)\in\overline{\mathcal{P}_{\rm temp}}} \sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}} |\alpha_y^{k,i}|^2 = Kn|\beta_{\mathcal{P}_{\rm temp},1}|^2 \ge \frac{1}{2} \tag{76}$$

which follows from (72).

Combining the above discussions, we see that, if $\mathcal{P}_{\rm hard} \cap \overline{\mathcal{P}_{\rm temp}} \neq \emptyset$, we obtain an element in $\mathcal{P}_{\rm soft} \cap \overline{\mathcal{P}_{\rm temp}}$ by $\mathsf{QAA}(\tilde{O}^{\rm sc,QMCI}_{\epsilon',\delta',\mathcal{P}_{\rm temp}}, \frac{1}{2Kn}, \delta'')$ and the subsequent measurement on $|\xi_{\mathcal{P}_{\rm temp},1}\rangle$ with probability at least $(1-\frac{\delta}{2Kn})^2 \ge 1-\frac{\delta}{Kn}$. Therefore, with some probability, the following happens: we successively obtain elements in $\mathcal{P}_{\rm soft}$ in loop 2–12 in Algorithm 2, until we get all the elements in $\mathcal{P}_{\rm hard}$. Since the number of loops is at most $n_{\rm soft}$, the probability that this happens is at least

$$\left(1-\frac{\delta}{Kn}\right)^{n_{\rm soft}} \ge \left(1-\frac{\delta}{Kn}\right)^{Kn} \ge 1-\delta. \tag{77}$$

We can evaluate the query complexity in this loop as (28) and (29) under the current setting of ϵ', δ' and δ'', recalling that $\mathsf{QAA}\left(\tilde{O}^{\rm sc,QMCI}_{\epsilon',\delta',\mathcal{P}_{\rm temp}}, \frac{1}{2Kn}, \delta''\right)$ calls $\tilde{O}^{\rm sc,QMCI}_{\epsilon',\delta',\mathcal{P}_{\rm temp}}$ $O\left(\sqrt{Kn}\log\left(\frac{1}{\delta''}\right)\right)$ times and that $\tilde{O}^{\rm sc,QMCI}_{\epsilon',\delta',\mathcal{P}_{\rm temp}}$ calls $O_{\rm seq}$ and $O_{\rm PWM}$ $O\left(\frac{1}{\epsilon'}\log\left(\frac{1}{\delta'}\right)\right)$ times and $O_{\mathcal{P}}$ $O(1)$ times.

(ii) $n_{\rm soft} > 0$: With certainty, the first run of QAA outputs the message "failure" or, even if not, $w_{k,i}$ classically calculated at step 6 in Algorithm 2 is smaller than $w_{\rm soft}$, since every $(k,i)\in\mathcal{P}_{\rm all}$ yields $w_{k,i} < w_{\rm soft}$ in this case. Therefore, the algorithm certainly ends outputting "no match," with QAA run only once. The query complexity of this is evaluated as (30) and (31).

(iii) $n_{\rm hard} = 0$ *and* $n_{\rm soft} > 0$: The algorithm ends with only one QAA that outputs "failure" or $(k,i)\in\mathcal{P}_{\rm all}$ such that $w_{k,i} < w_{\rm soft}$, or QAA runs sometimes outputting $(k,i)\in\mathcal{P}_{\rm soft}$. The number of QAA loops is at most $n_{\rm soft}$, and thus, the query complexity is evaluated as (28) and (29) similarly to case (i). ■

Let us make a comment on the scaling of the complexity bounds (28) and (29) on $n_{\rm soft}$. Note that, although (18) and (20) scale with $n_{\rm sol}$ as $O(\sqrt{n_{\rm sol}})$, (28) and (29) scale linearly with $n_{\rm soft}$, which means that the QMCI-based method has the worse scaling with respect to the number of matches. This difference can be understood as follows. In the naive iteration method, among the computational basis states contained in the state $|\Psi_{\mathcal{P}_{\rm temp}}\rangle$, those with the last qubit taking $|1\rangle$, are $|k\rangle|i\rangle|w_{k,i}\rangle|w_{\rm th}\rangle|1\rangle|1\rangle|1\rangle$ with $(k,i)\in\mathcal{P}_{\rm sol}\cap\overline{\mathcal{P}_{\rm temp}}$, each of which having the amplitude $\sqrt{\frac{1}{Kn}}$. They constitute $|\psi_{\mathcal{P}_{\rm temp},1}\rangle|1\rangle$, the target state of QAA, whose amplitude decreases as $\sqrt{\frac{n_{\rm sol}}{Kn}}, \sqrt{\frac{n_{\rm sol}-1}{Kn}}, \dots, \sqrt{\frac{1}{Kn}}$ in the QAA loop, and this leads to the evaluation of the total complexity in (28) and (29). On the other hand, in the QMCI-based method, among the computational basis states contained in the state $|\Xi_{\mathcal{P}_{\rm temp}}\rangle$, those with the last qubit taking $|1\rangle$ are $|k\rangle|i\rangle|y\rangle|\frac{w_{\rm th}}{m}\rangle|1\rangle|1\rangle|1\rangle$, with (k,i) being any elements in $\overline{\mathcal{P}_{\rm temp}}$, although those for $(k,i)\in\mathcal{P}_{\rm soft}$ constitute the most part of $|\Xi_{\mathcal{P}_{\rm temp}}\rangle$ in terms of the squared amplitude. When we write $|\Xi_{\mathcal{P}_{\rm temp}}\rangle$ as

$$|\Xi_{\mathcal{P}_{\rm temp}}\rangle = \frac{1}{\sqrt{Kn}}\sum_{(k,i)\in\overline{\mathcal{P}_{\rm temp}}} \gamma_{k,i}|\xi_{\mathcal{P}_{\rm temp},1;k,i}\rangle + \beta_{\mathcal{P}_{\rm temp},0}|\xi_{\mathcal{P}_{\rm temp},0}\rangle|0\rangle \tag{78}$$

with

$$|\tilde{\xi}_{\mathcal{P}_{\rm temp},1;k,i}\rangle = \frac{1}{\gamma_{k,i}} |k\rangle\,|i\rangle \sum_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}} \alpha_y^{k,i}\,|y\rangle \left|\frac{w_{\rm th}}{m}\right\rangle |1\rangle\,|1\rangle\,|1\rangle$$

$$\gamma_{k,i} = \sqrt{\sum\nolimits_{\substack{y\in\mathcal{Y}_{k,i}\\ y\ge w_{\rm mid}/m}} \left|\alpha_y^{k,i}\right|^2} \tag{79}$$

the squared amplitude of $|\tilde{\xi}_{\mathcal{P}_{\rm temp},1;k,i}\rangle$, $\frac{|\gamma_{k,i}|^2}{Kn}$, is at least $\frac{1}{2Kn}$ for $(k,i)\in\mathcal{P}_{\rm hard}$ as we see from (71), but that for $(k,i)\in\mathcal{P}_{\rm soft}\backslash\mathcal{P}_{\rm hard}$ can be much smaller than it. Nevertheless, the squared amplitudes of the states $|\tilde{\xi}_{\mathcal{P}_{\rm temp},1;k,i}\rangle$ for $(k,i)\in\mathcal{P}_{\rm soft}\backslash\mathcal{P}_{\rm hard}$ can pile up to the value comparable with $\frac{1}{2Kn}$. In such a situation, it is possible that, in the QAA loop, QAA $\left(\tilde{O}^{\rm sc,QMCI}_{\epsilon',\delta',\mathcal{P}_{\rm temp}}, \frac{1}{2Kn}, \delta''\right)$ continues to output $|\xi_{\mathcal{P}_{\rm temp},1}\rangle$, and we continue to get $(k,i)\in\mathcal{P}_{\rm soft}$, until we get $O(n_{\rm soft})$ elements in $\mathcal{P}_{\rm soft}$ and the squared amplitude $|\beta_{\mathcal{P}_{\rm temp},1}|^2$ of $|\xi_{\mathcal{P}_{\rm temp},1}\rangle\,|1\rangle$ in $|\Xi_{\mathcal{P}_{\rm temp}}\rangle$ decreases below $\frac{1}{2Kn}$. When this happens, the query complexity becomes comparable with the bounds (28) and (29).

REFERENCES

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition.*, Cambridge, U.K.: Cambridge Univ. Press, 2010.

[2] G. D. Stormo, T. D. Schneider, L. Gold, and A. Ehrenfeucht, "Use of the 'perceptron' algorithm to distinguish translational initiation sites in E. coli," *Nucleic Acids Res.*, vol. 10, no. 9, pp. 2997–3011, 1982, doi: 10.1093/nar/10.9.2997.

[3] M. Gribskov, A. D. McLachlan, and D. Eisenberg, "Profile analysis: Detection of distantly related proteins," *Proc. Nat. Acad. Sci.*, vol. 84, no. 13, pp. 4355–4358, 1987, doi: 10.1073/pnas.84.13.4355.

[4] K. Quandt, K. Frech, H. Karas, E. Wingender, and T. Werner, "Matlnd and Matlnspector: New fast and versatile tools for detection of consensus matches in nucleotide sequence data," *Nucleic Acids Res.*, vol. 23, no. 23, pp. 4878–4884, 1995, doi: 10.1093/nar/23.23.4878.

[5] T. D. Wu, C. G. Nevill-Manning, and D. L. Brutlag, "Fast probabilistic analysis of sequence function using scoring matrices," *Bioinformatics*, vol. 16, no. 3, pp. 233–244, 2000, doi: 10.1093/bioinformatics/16.3.233.

[6] B. Dorohonceanu and C. G. Nevill-Manning, "Accelerating protein classification using suffix trees," in *Proc. 8th Int. Conf. Intell. Syst. Mol. Biol.*, 2000, pp. 128–133. [Online]. Available: https://aaai.org/proceeding/01-ismb-2000/

[7] S. Rajasekaran, X. Jin, and J. L. Spouge, "The efficient computation of position-specific match scores with the fast Fourier transform," *J. Comput. Biol.*, vol. 9, no. 1, pp. 23–33, 2002, doi: 10.1089/10665270252833172.

[8] A. E. Kel, E. Gößling, I. Reuter, E. Cheremushkin, O. V. Kel-Margoulis, and E. Wingender, "MATCHTM: A tool for searching transcription factor binding sites in DNA sequences," *Nucleic Acids Res.*, vol. 31, no. 13, pp. 3576–3579, 2003, doi: 10.1093/nar/gkg585.

[9] M. Beckstette, D. Strothmann, R. Homann, R. Giegerich, and S. Kurtz, "PoSSuMsearch: Fast and sensitive matching of position specific scoring matrices using enhanced suffix arrays," in *Proc. German Conf. Bioinf.*, 2004, pp. 53–64. [Online]. Available: https://dl.gi.de/items/f09ccca6-9233-46bd-acdf-62df3a6a7f5a

[10] V. Freschi and A. Bogliolo, "Using sequence compression to speedup probabilistic profile matching," *Bioinformatics*, vol. 21, no. 10, pp. 2225–2229, 2005, doi: 10.1093/bioinformatics/bti323.

[11] M. Beckstette, R. Homann, R. Giegerich, and S. Kurtz, "Fast index based algorithms and software for matching position specific scoring matrices," *BMC Bioinf.*, vol. 7, no. 1, pp. 1–25, 2006, doi: 10.1186/1471-2105-7-389.

[12] A. Liefooghe, H. Touzet, and J.-S. Varré, "Large scale matching for position weight matrices," in *Combinatorial Pattern Matching*, M. Lewenstein and G. Valiente, Eds., Berlin, Germany: Springer, 2006, pp. 401–412.

[13] C. Pizzi, P. Rastas, and E. Ukkonen, "Fast search algorithms for position specific scoring matrices," in *Bioinformatics Research and Development*, S. Hochreiter and R. Wagner, Eds., Berlin, Germany: Springer, 2007, pp. 239–250.

[14] J. Korhonen, P. Martinmäki, C. Pizzi, P. Rastas, and E. Ukkonen, "MOODS: Fast search for position weight matrix matches in DNA sequences," *Bioinformatics*, vol. 25, no. 23, pp. 3181–3182, 2009, doi: 10.1093/bioinformatics/btp554.

[15] C. Pizzi, P. Rastas, and E. Ukkonen, "Finding significant matches of position weight matrices in linear time," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 8, no. 1, pp. 69–79, Jan./Feb. 2011, doi: 10.1109/TCBB.2009.35.

[16] J. Fostier, "BLAMM: BLAS-based algorithm for finding position weight matrix occurrences in DNA sequences on CPUs and GPUs," *BMC Bioinf.*, vol. 21, no. 2, pp. 1–13, 2020, doi: 10.1186/s12859-020-3348-6.

[17] H. Ramesh and V. Vinay, "String matching in O(n+m) quantum time," *J. Discrete Algorithms*, vol. 1, no. 1, pp. 103–110, 2003, doi: 10.1016/S1570-8667(03)00010-8.

[18] A. Montanaro, "Quantum pattern matching fast on average," *Algorithmica*, vol. 77, no. 1, pp. 16–39, 2017, doi: 10.1007/s00453-015-0060-4.

[19] P. Niroula and Y. Nam, "A quantum algorithm for string matching," *npj Quantum Inf.*, vol. 7, no. 1, 2021, Art. no. 37, doi: 10.1038/s41534-021-00369-3.

[20] K. K. Soni and A. Rasool, "Quantum-based exact pattern matching algorithms for biological sequences," *ETRI J.*, vol. 43, no. 3, pp. 483–510, 2021, doi: 10.4218/etrij.2019-0589.

[21] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," *Contemporary Math.*, vol. 305, pp. 53–74, 2002, doi: 10.1090/conm/305/05215.

[22] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput.*, 1996, pp. 212–219, doi: 10.1145/237814.237866.

[23] A. Montanaro, "Quantum speedup of Monte Carlo methods," *Proc. Roy. Soc. Ser. A*, vol. 471, no. 2181, 2015, Art. no. 20150301, doi: 10.1098/rspa.2015.0301.

[24] K. Miyamoto, G. Morrás, T. S. Yamamoto, S. Kuroyanagi, and S. Nesseris, "Gravitational wave matched filtering by quantum Monte Carlo integration and quantum amplitude amplification," *Phys. Rev. Res.*, vol. 4, 2022, Art. no. 033150, doi: 10.1103/PhysRevResearch.4.033150.

[25] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," *Phys. Rev. Lett.*, vol. 100, Apr. 2008, Art. no. 160501, doi: 10.1103/PhysRevLett.100.160501.

[26] K. Shameer et al., "3PFDB-A database of best representative PSSM profiles (BRPS) of protein families generated using a novel data mining approach," *BioData Mining*, vol. 2, no. 1, pp. 1–10, 2009, doi: 10.1186/1756-0381-2-8.

[27] E. M. Coreas and H. Thapliyal, "Everything you always wanted to know about quantum circuits," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, J.G. Webster, Ed. Hoboken, NJ, USA: Wiley, 2022, pp. 1–17, doi: 10.1002/047134608X.W8440.

[28] I. Koren, *Computer Arithmetic Algorithms.*, Natick, MA, USA: AK Peters/CRC Press, 2001, doi: 10.1201/9781315275567.

[29] L. Grover and T. Rudolph, "Creating superpositions that correspond to efficiently integrable probability distributions," 2002, *arXiv:quant-ph/0208112*, doi: 10.48550/arXiv.quant-ph/0208112.

[30] S. Arunachalam, V. Gheorghiu, T. Jochym-O'Connor, M. Mosca, and P. V. Srinivasan, "On the robustness of bucket brigade quantum RAM," *New J. Phys.*, vol. vol. 17, no. 12, Dec. 2015, Art. no. 123010, doi: 10.1088/1367-2630/17/12/123010.

[31] P. Billingsley, *Probability and Measure*. Hoboken, NJ, USA: Wiley, 2008.

[32] Y. LeCun et al., "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory Neural Networks*. Cambridge, MA, USA: MIT Press, 1995.

[33] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon et al., Eds. vol. 30. Red Hook, NY, USA: Curran Associates, 2017.

[34] M. R. Jerrum, L. G. Valiant, and Vijay V. Vazirani, "Random generation of combinatorial structures from a uniform distribution," *Theor. Comput. Sci.*, vol. 43, pp. 169–188, 1986, doi: 10.1016/0304-3975(86)90174-X.