Algorithms **2013**, *6*, 636-677; doi:10.3390/a6040636

OPEN ACCESS

algorithms

ISSN 1999-4893

www.mdpi.com/journal/algorithms

Article

Sublinear Time Motif Discovery from Multiple Sequences

Bin Fu *, Yunhui Fu and Yuan Xue

Department of Computer Science, University of Texas-Pan American, 1201 W University Dr.,
Edinburg, TX 78539, USA; E-Mails: fuyunhui@gmail.com (Y.F.); xuey@utpa.edu (Y.X.)

* Author to whom correspondence should be addressed; E-Mail: bfu@utpa.edu;
Tel.: +1-956-665-3635; Fax: +1-956-665-5099.

*Received: 11 June 2013; in revised form: 30 September 2013 / Accepted: 1 October 2013/
Published: 14 October 2013*

Abstract: In this paper, a natural probabilistic model for motif discovery has been used to experimentally test the quality of motif discovery programs. In this model, there are k background sequences, and each character in a background sequence is a random character from an alphabet, Σ. A motif $G = g_1g_2\ldots g_m$ is a string of m characters. In each background sequence is implanted a probabilistically-generated approximate copy of G. For a probabilistically-generated approximate copy $b_1b_2\ldots b_m$ of G, every character, b_i, is probabilistically generated, such that the probability for $b_i \neq g_i$ is at most α. We develop two new randomized algorithms and one new deterministic algorithm. They make advancements in the following aspects: (1) The algorithms are much faster than those before. Our algorithms can even run in sublinear time. (2) They can handle any motif pattern. (3) The restriction for the alphabet size is a lower bound of four. This gives them potential applications in practical problems, since gene sequences have an alphabet size of four. (4) All algorithms have rigorous proofs about their performances. The methods developed in this paper have been used in the software implementation. We observed some encouraging results that show improved performance for motif detection compared with other software.

Keywords: motif discovery; sublinear time; randomized algorithm; deterministic algorithm

1. Introduction

Motif discovery is an important problem in computational biology and computer science. For instance, it has applications in coding theory [1,2], locating binding sites and conserved regions in unaligned sequences [3–6], genetic drug target identification [7], designing genetic probes [7] and universal PCR primer design [7–10].

This paper focuses on the application of motif discovery to find conserved regions in a set of given DNA, RNA or protein sequences. Such conserved regions may represent common biological functions or structures. Many performance measures have been proposed for motif discovery. Let C be a subset of 0–1 sequences of length n. The *covering radius* of C is the smallest integer, r, such that each vector in $\{0,1\}^n$ is at a distance at most r from a string in C. The decision problem associated with the covering radius for a set of binary sequences is NP-complete [1]. The similar closest string and substring problems were proven to be NP-hard [1,7]. Some approximation algorithms have been proposed. Li *et al.* [11] gave an approximation scheme for the closest string and substring problems. The related consensus patterns problem is that given n sequences $s_1, \cdots, s_n$, find a region of length L in each s_i and a string, s, of length L, so that the total Hamming distance from s to these regions is minimized. Approximation algorithms for the consensus patterns problem were reported in [12]. Furthermore, a number of heuristics and programs have been developed [13–17].

In many applications, motifs are faint and may not be apparent when two sequences alone are compared, but may become clearer when more sequences are compared together [18]. For this reason, it has been conjectured that comparing more sequences together can help with identifying faint motifs. This paper is a theoretical approach with a rigorous probabilistic analysis.

We study a natural probabilistic model for motif discovery. In this model, there are k background sequences, and each character in the background sequence is a random character from an alphabet, Σ. A motif $G = g_1 g_2 \ldots g_m$ is a string of m characters. In each background sequence is implanted a probabilistically-generated approximate copy of G. For a probabilistically-generated approximate copy $b_1 b_2 \ldots b_m$ of G, every character, b_i, is probabilistically generated, such that the probability for $b_i \neq g_i$, which is called a *mutation*, is at most α. This model was first proposed in [13] and has been widely used in experimentally testing motif discovery programs [14–17]. We note that a mutation in our model converts a character, g_i, in the motif into a different character, b_i, without probability restriction. This means that a character, g_i, in the motif may not become any character b_i in $\Sigma - \{g_i\}$ with equal probability.

We develop three algorithms for which, under the probabilistic model, one can find the implanted motif with high probability via a tradeoff between computational time and the probability of mutation. Each algorithm has a preprocessing phase and the voting phase. We use a pair of functions, $(t_1(n,k), t_2(n,k))$, to describe the computational complexity of the motif detection algorithm, where n is the largest length of the input sequence and k is the number of sequences. Function $t_1(n,k)$ is the time complexity for the part for preprocessing, and $t_2(n,k)$ is the time complexity for recovering one character for the motif after preprocessing. The total time is $O(t_1(n,k) + t_2(n,k)|G|)$.

(1) There exists a randomized algorithm, such that there are positive constants, c_0 and c_1, such that, if the alphabet size is at least four, the number of sequences is at least $c_1 \log n$, the motif length is at least $c_0 \log n$ and each character in the motif region has a probability of at most $\frac{1}{(\log n)^{2+\mu}}$ of mutation for some fixed $\mu > 0$, then the motif can be recovered with a probability of at least $\frac{3}{4}$ in $(O(\frac{n}{\sqrt{h}}(\log n)^{\frac{7}{2}} + h^2 \log^2 n), O(\log n))$ time, where n is the longest length of any input sequences and $h = \min(|G|, n^{\frac{2}{5}})$. The algorithm's total time is sublinear if the motif length, $|G|$, is in the range $[(\log n)^{7+\mu}, \frac{n}{(\log n)^{1+\mu}}]$. This is the first sublinear time algorithm with rigorous analysis in this model.

(2) There exists a randomized algorithm, such that there are positive constants, c_0, c_1 and α, such that if the alphabet size is at least four, the number of sequences is at least $c_1 \log n$, the motif length is at least $c_0 \log n$ and each character in the motif region has a probability of at most α of mutation, then the motif can be recovered with a probability of at least $\frac{3}{4}$ in $(O(\frac{n^2}{|G|}(\log n)^{O(1)}), O(\log n))$ time.

(3) There exists a deterministic algorithm, such that there are positive constants, c_0, c_1 and α, such that if the alphabet size is at least four, the number of sequences is at least $c_1 \log n$, the motif length is at least $c_0 \log n$ and each character in the motif region has a probability of at most α of mutation, then the motif can be recovered with a probability of at least $\frac{3}{4}$ in $(O(n^2(\log n)^{O(1)}), O(\log n))$ time.

The research in this model has been reported in [19–21]. In [19], Fu *et al.*. developed an algorithm that needs the alphabet size to be a constant that is much larger than four. In [20], our algorithm cannot handle all possible motif patterns. In [21], Liu *et al.* designed an algorithm that runs in $O(n^3)$ time and lacks rigorous analysis about its performance. The motif recovery in this natural and simple model has not been fully understood and seems to be a complicated problem.

This paper presents two new randomized algorithms and one new deterministic algorithm. They make advancements in the following aspects: (1) The algorithms are much faster than those before. Our algorithms can even run in sublinear time. (2) They can handle any motif pattern. (3) The restriction for the alphabet size is as small as four, giving them potential applications in practical problems, since gene sequences have an alphabet size of four. (4) All algorithms have rigorous proofs about their performances.

The algorithm for motif detection is named Recover-Motif(.). The entire Recover-Motif(.) is described in Section 4.2. We analyze Algorithm Recover-Motif (.) in Section 6.

2. Notations and the Model of Sequence Generation

For a set, A, $||A||$ denotes the number of elements in A. Σ is an alphabet with $||\Sigma|| = t \geq 2$. For an integer, $n \geq 0$, Σ^n is the set of sequences of a length of n with characters from Σ. For a sequence $S = a_1a_2\cdots a_n$, $S[i]$ denotes the character, a_i, and $S[i,j]$ denotes the substring, $a_i\cdots a_j$, for $1 \leq i \leq j \leq n$. $|S|$ denotes the length of the sequence, S. We use $\emptyset$ to represent the empty sequence, which has a length of zero.

Let $G = g_1g_2\cdots g_m$ be a fixed sequence of m characters. G is the motif to be discovered by our algorithm. A $\Theta(n, G, \alpha)$-sequence has the form $S = a_1\cdots a_{n_1}b_1\cdots b_m a_{n_1+1}\cdots a_{n_2}$, where $n_2+m \leq n$, each a_i has a probability of $\frac{1}{t}$ to be equal to π for each $\pi \in \Sigma$ and b_i has a probability of at most α and not equal to g_i for $1 \leq i \leq m$, where $m = |G|$. $\aleph(S)$ denotes the motif region $b_1\cdots b_m$ of S. A mutation converts a character, g_i, in the motif into an arbitrary, different character, b_i, without probability restriction. This allows a character, g_i, in the motif to change into any character, b_i, in $\Sigma - \{g_i\}$ with even a different probability. The motif region of S may start at an arbitrary position in S. Furthermore, a mutation may convert a character, g_i, in the motif into an arbitrary, different character, b_i, only subject to the restriction that g_i will mutate with a probability of at most α.

For two sequences $S_1 = a_1\cdots a_m$ and $S_2 = b_1\cdots b_m$ of the same length, let the *relative Hamming distance* $\text{diff}(S_1, S_2) = \frac{|\{i|a_i\neq b_i(i=1,\cdots,m)\}|}{m}$.

Definition 1. For two intervals, $[i_1, j_1]$ and $[i_2, j_2]$, define $\text{shift}([i_1, j_1], [i_2, j_2]) = \min(|i_1-i_2|, |j_1-j_2|)$.

3. Brief Introduction to the Algorithm

Every detection algorithm in this paper has two phases. The first phase is preprocessing, so that the motif regions from multiple sequences can be aligned in the same column region. The second phase is to recover the motif via voting. We use a pair of functions, $(t_1(n,k), t_2(n,k))$, to describe the computational complexity of the motif detection algorithm. Function $t_1(n,k)$ is the time complexity for the preprocessing phase, and $t_2(n,k)$ is the time complexity for outputting one character for the motif in the voting phase.

The motif, G, is a pattern unknown to algorithm Recover-Motif, and algorithm Recover-Motif will attempt to recover G from a series of $\Theta(n, G, \alpha)$-sequences generated by the probabilistic model.

3.1. Algorithm

The algorithm first detects a position that is close to the left motif boundary in a sequence. It finds such a position via sampling and collision between two sequences. After the rough left boundary of a sequence is found, it is used to find the rough boundaries of the rest of the sequences. Similarly, we find those right boundaries of the motif among the input sequences. The exact left boundary of each motif region will be detected in the next phase via voting. Each character of the motif is recovered by voting among all the characters at the same positions in the motif regions of input sequences. For a sequence, S, a sample point is a random position, i, in S. For two sequences, S and S', with two sample points, i and j, respectively, a rough motif boundary is detected by the similarity of $S[i, i+l]$ and $S'[i, i+l]$ for some reasonably large parameter, l.

Descriptions of the Algorithm

```
Input: $Z = Z_1 \cup Z_2$, where $Z_1 = \{S'_1, \cdots, S'_{2k_1}\}$ and $Z_2 = \{S''_1, \cdots, S''_{k_2}\}$ are two sets of input sequences.
Output: planted motif in each sequence and consensus string. Start:
Randomly select sample points from each sequence, both in $Z_1$ and $Z_2$.
For each pair of sequences selected from $Z_1$ and $Z_2$,
    find the rough left and rough right boundaries via the matching at sample points.
    Improve the rough boundaries.
If the motif boundaries of each sequence in $Z_2$ are not empty,
    use the Voting algorithm to get the planted motifs.
End of Algorithm.
```

3.2. An Example

We provide the following example for the brief idea of our algorithm. Let the following input strings be defined as below. We assume that the original motif is **TTTTTAACGATTAGCS**. The motif part is displayed with bold font, and the mutated characters in the motif region have been marked by $*$ in their feet.

3.2.1. . Input Sequences

This contains two groups $Z_1 = \{S'_1, S'_2\}$ and $Z_2 = \{S''_1, S''_2, S''_3, S''_4, S''_5\}$.

$$\begin{aligned} &Z_1 : \\ S'_1 &= GTACCATGGA\mathbf{TTA_*TTAACGATTAGCS}TAGAGGACCTA. \\ S'_2 &= AATCCTTA\mathbf{C_*TTTTAACGATTAGCS}GTC. \end{aligned}$$

The above two strings are used to detect the initial motif region and use them to deal with the motif in the second group below.

$$\begin{aligned} &Z_2 : \\ S''_1 &= ATTCGATCCAG\mathbf{TTTTTAACGG_*TTAGCS}CAATTACTTAG. \\ S''_2 &= GCATTGCAT\mathbf{TTTTTAACGATTAC_*CS}GTACTTAGCTAGATC. \\ S''_3 &= TCAGGGCATCGAGAC\mathbf{TTTTTAG_*CGATTAGCS}CTAGAATCAGACCT. \\ S''_4 &= GTACCTGGCATTGAACG\mathbf{TTTTTAACGATTAGCA_*}TGCAGATGGACCTTTA. \\ S''_5 &= AATGGATCAGA\mathbf{TTTTTAACGATTC_*GCS}CTAGATTCAG. \end{aligned}$$

3.2.2. . Select Sample Points

Some sample points of two sequences in Z_1 are selected randomly and marked with the little dots on the top.

$$\begin{aligned} S'_1 &= GT\dot{A}CC\dot{A}TG\dot{G}A\mathbf{T\dot{T}A_*TTA\dot{A}CGA\dot{T}T\dot{A}GC\dot{S}}TA\dot{G}AG\dot{G}ACC\dot{T}A. \\ S'_2 &= \dot{A}AT\dot{C}CTT\dot{A}\mathbf{C_*\dot{T}TTT\dot{A}AC\dot{G}A\dot{T}TA\dot{G}CS}\dot{G}TC. \end{aligned}$$

3.2.3. . Collision Detection

In this step, the left and right rough boundaries of two sequences will be marked. The following shows the left collision, which happens nearby the left motif boundary and are marked by two overline $\overline{TATT}$ and $\overline{TTTT}$ subsequences.

$$\begin{aligned} S'_1 &= GT\dot{A}CC\dot{A}TG\dot{G}A\mathbf{\overline{T\dot{T}A_*TT}A\dot{A}CGA\dot{T}T\dot{A}GC\dot{S}}TA\dot{G}AG\dot{G}ACC\dot{T}A. \\ S'_2 &= \dot{A}AT\dot{C}C_*TT\dot{A}\mathbf{C_*\overline{\dot{T}TTT}\dot{A}AC\dot{G}A\dot{T}TA\dot{G}CS}\dot{G}TC. \end{aligned}$$

The following shows the right collision, which happens nearby the right motif boundary and is marked by two overline $\overline{TTAG}$ subsequences.

$$\begin{aligned} S'_1 &= GT\dot{A}CC\dot{A}_*TG\dot{G}A\mathbf{T\dot{T}A_*TTA\dot{A}CGA\overline{\dot{T}T\dot{A}G}C\dot{S}}TA\dot{G}AG\dot{G}ACC\dot{T}A. \\ S'_2 &= \dot{A}AT\dot{C}C_*TT\dot{A}\mathbf{C_*\dot{T}TTT\dot{A}AC\dot{G}A\overline{\dot{T}TA\dot{G}}CS}\dot{G}TC. \end{aligned}$$

3.2.4. . Improving the Boundaries

In the early phase of the algorithm, we first detect a small piece of the motif in S'_1 by comparing S'_1 and S'_2. Assume that "$\mathbf{TA_*TT}$" and "**TTAG**" are found in the left and right motif regions of

S'_1, respectively. The rough motif length will be calculated via the difference of the location of the first character, 'T', of the first subsequence and the location of the last character, 'G', of the second subsequence. The position marked by "$\underline{\text{A}}$" is the rough left boundary of the motif, and the position marked by "$\underline{\text{T}}$" is the rough right boundary of the motif in S'_1 below.

$$S'_1 = GTACCATGG\underline{A}\mathbf{TTA_*TTAACGATTAGCS}\underline{T}AGAGGACCTA.$$
$$S'_2 = AATCCTT\underline{A}\mathbf{C_*TTTTAACGATTAGCS}\underline{G}TC.$$

3.2.5. . Select Sample Points for the Sequences in Z_2

Some sample points near the motif boundaries of S'_1 are selected.
$S'_1 = GTACCATG\dot{G}A\mathbf{T\dot{T}A_*\dot{T}TAACGATT\dot{A}G\dot{C}S}T\dot{A}GAGGACCTA.$
Sample points are selected in each sequence in Z_2.

$$S''_1 = A\dot{T}TC\dot{G}ATCC\dot{A}G\mathbf{T\dot{T}T\dot{T}TAACGG_*TTAG\dot{C}S}C\dot{A}AT\dot{T}ACTT\dot{A}G.$$
$$S''_2 = G\dot{C}ATT\dot{G}CAT\mathbf{T\dot{T}TTTAACGATTAC_*\dot{C}S}GT\dot{A}CTT\dot{A}GCT\dot{A}GA\dot{T}C.$$
$$S''_3 = \dot{T}CA\dot{G}GGCA\dot{T}CGA\dot{G}AC\mathbf{TTT\dot{T}TAG_*CGATTAG\dot{C}S}CTA\dot{G}AATC\dot{A}GAC\dot{C}T.$$
$$S''_4 = GT\dot{A}CCT\dot{G}GCAT\dot{T}GAACG\mathbf{T\dot{T}TTTAACGATT\dot{A}GCA_*}TGC\dot{A}GAT\dot{G}GACCT\dot{T}TA.$$
$$S''_5 = AA\dot{T}GGA\dot{T}CAGA\mathbf{T\dot{T}TTTAACGATTC_*G\dot{C}S}CTA\dot{G}ATT\dot{C}AG.$$

3.2.6. . Collision Detection Between S'_1 with the Sequences in Z_2

The rough motif boundaries of the sequences in Z_2 are detected via the collisions with the subsequences near the motif area of S'_1.
$S'_1 = GTACCATG\dot{G}A\mathbf{T\dot{T}A_*\dot{T}TAACGATT\dot{A}G\dot{C}S}T\dot{A}GAGGACCTA.$
The rough motif boundaries are marked by the lines over the matched subsequences.

$$S''_1 = A\dot{T}TC\dot{G}ATCC\dot{A}G\mathbf{T\overline{\dot{T}T\dot{T}T}AACGG_*T\overline{TAG\dot{C}}S}C\dot{A}AT\dot{T}ACTT\dot{A}G.$$
$$S''_2 = G\dot{C}ATT\dot{G}CAT\mathbf{T\overline{\dot{T}TTT}AACGAT\overline{TAC_*\dot{C}}S}GT\dot{A}CTT\dot{A}GCT\dot{A}GA\dot{T}C.$$
$$S''_3 = \dot{T}CA\dot{G}GGCA\dot{T}CGA\dot{G}AC\mathbf{TTT\overline{\dot{T}TAG_*}CGAT\overline{TAG\dot{C}}S}CTA\dot{G}AATC\dot{A}GAC\dot{C}T.$$
$$S''_4 = GT\dot{A}CCT\dot{G}GCAT\dot{T}GAACG\mathbf{T\overline{\dot{T}TTT}AACG\overline{ATT\dot{A}}GCA_*}TGC\dot{A}GAT\dot{G}GACCT\dot{T}TA.$$
$$S''_5 = AA\dot{T}GGA\dot{T}CAGA\mathbf{T\overline{\dot{T}TTT}AACGAT\overline{TC_*G\dot{C}}S}CTA\dot{G}ATT\dot{C}AG.$$

3.2.7. . Improving the Motif Boundaries for the Sequences in Z_2

After the collision with the sequences in Z_2, we obtain the rough location of the motifs of the sequences in Z_2. Their rough motif boundaries for the sequences in Z_2 are improved to be closer to exact boundaries.
$S'_1 = GTACCATGGA\mathbf{\overline{TTA_*T}TAACGATT\overline{AGCS}}TAGAGGACCTA.$

The improved motif boundaries of the sequences in Z_2 are marked below. This phase does not cost much time, as only the positions near the rough motif boundaries are tested.

$$\begin{aligned}
S_1'' &= ATTCGATCCA\underline{G}\mathbf{TTTTTAACGG}_*\mathbf{TTAGCS}\underline{C}AATTACTTAG.\\
S_2'' &= GCATTGC\underline{A}T\mathbf{TTTTTAACGATTAC}_*\mathbf{CS}\underline{G}TACTTAGCTAGATC.\\
S_3'' &= TCAGGGCATCGAGA\underline{C}\mathbf{TTTTTAG}_*\mathbf{CGATTAGCS}\underline{C}TAGAATCAGACCT.\\
S_4'' &= GTACCTGGCATTGAAC\underline{G}\mathbf{TTTTTAACGATTAGCA}_*\underline{T}GCAGATGGACCTTTA.\\
S_5'' &= AATGGATCAG\underline{A}\mathbf{TTTTTAACGATTC}_*\mathbf{GCS}C\underline{T}AGATTCAG.
\end{aligned}$$

3.2.8. . Motif Boundaries for the Sequences in Z_2

$S_1' = GTACCATGGA\overline{\mathbf{TTA}_*\mathbf{T}}\mathbf{TAACGATT}\overline{\mathbf{AGCS}}TAGAGGACCTA.$

Use the pair (G_L, G_R) with $G_L = \overline{\mathbf{TTAT}}$ and $G_R = \overline{\mathbf{AGCS}}$ to find the motif boundaries in the sequences of Z_2. The rough boundaries of the second group are marked below with underlines. In this phase, the exact motif boundaries for most sequences of Z_2 can be found.

$$\begin{aligned}
S_1'' &= ATTCGATCCA\underline{G}\mathbf{TTTTTAACGG}_*\mathbf{TTAGCS}\underline{C}AATTACTTAG.\\
S_2'' &= GCATTGCA\underline{T}\mathbf{TTTTTAACGATTAC}_*\mathbf{CS}\underline{G}TACTTAGCTAGATC.\\
S_3'' &= TCAGGGCATCGAGA\underline{C}\mathbf{TTTTTAG}_*\mathbf{CGATTAGCS}\underline{C}TAGAATCAGACCT.\\
S_4'' &= GTACCTGGCATTGAAC\underline{G}\mathbf{TTTTTAACGATTAGCA}_*\underline{T}GCAGATGGACCTTTA.\\
S_5'' &= AATGGATCAG\underline{A}\mathbf{TTTTTAACGATTC}_*\mathbf{GCS}\underline{C}TAGATTCAG.
\end{aligned}$$

3.2.9. . Extracting the Motif Regions

The motif regions of the second group will be extracted. The original motif is recovered via voting at each column.

$$\begin{aligned}
G_1'' &= \mathbf{TTTTTAACGG}_*\mathbf{TTAGCS}\\
G_2'' &= \mathbf{TTTTTAACGATTAC}_*\mathbf{CS}\\
G_3'' &= \mathbf{TTTTTAG}_*\mathbf{CGATTAGCS}\\
G_4'' &= \mathbf{TTTTTAACGATTAGCA}_*\\
G_5'' &= \mathbf{TTTTTAACGATTC}_*\mathbf{GCS}
\end{aligned}$$

3.2.10. . Recovering Motif via Voting

The original motif, **TTTTTAACGATTAGCS**, is recovered via voting at all columns. For example, the last **S** in the motif is recovered via voting among the characters, **S**, **S**, **S**, **A**, **S**, in the last column.

3.3. Our Results

We give an algorithm for the case with at most a $\frac{1}{(\log n)^{2+\mu}}$ mutation rate. The performance of the algorithm is stated in Theorem 2. Theorem 2 implies Corollary 3 by selecting $k = c_1 \log n$ with some constant c_1 that is large enough.

Theorem 2. *Assume that μ is a fixed number in $(0,1)$ and the alphabet size, t, is at least four. There exists a randomized algorithm and a constant c_0, such that if the length of the motif, G, is at least $c_0 \log n$, then, given k independent $\Theta(n, G, \frac{1}{(\log n)^{2+\mu}})$-sequences, the algorithm outputs G', such that:*

(1) with a probability of at most $e^{-\Omega(k)}$, $|G'| \neq |G|$;

(2) for each $1 \leq i \leq |G|$, with a probability of at most $e^{-\Omega(k)}$, $G'[i] \neq G[i]$; and

(3) with a probability of at most $\frac{k}{n^3}$, the algorithm does not stop in $(O(k(\frac{n}{\sqrt{h}}(\log n)^{\frac{5}{2}} + h^2 \log n)), O(k))$ time, where n is the longest length of any input sequences, and $h = \min(|G|, n^{\frac{2}{5}})$.

Corollary 3. *There exists a randomized algorithm and positive constants, c_0, c_1 and μ, such that if the alphabet size is at least four, the number of sequences is at least $c_1 \log n$, the motif length is at least $c_0 \log n$ and each character in the motif region has a probability of at most $\frac{1}{(\log n)^{2+\mu}}$ of mutation, then the motif can be recovered with a probability of at least $\frac{3}{4}$ in $(O(\frac{n}{\sqrt{h}}(\log n)^{\frac{7}{2}} + h^2 \log^2 n), O(\log n))$ time, where n is the longest length of any input sequences, and $h = \min(|G|, n^{\frac{2}{5}})$.*

We give a randomized algorithm for the case with a $\Omega(1)$ mutation rate. The performance of the algorithm is stated in Theorem 4. Theorem 4 implies Corollary 5 by selecting $k = c_1 \log n$ with some constant c_1 that is large enough.

Theorem 4. *Assume that the alphabet size, t, is at least four. There exists a randomized algorithm and a constant c_0, such that if the length of the motif G is at least $c_0 \log n$, then, given k independent $\Theta(n, G, \mu))$-sequences, the algorithm outputs G', such that:*

(1) with a probability of at most $e^{-\Omega(k)}$, $|G'| \neq |G|$;

(2) for each $1 \leq i \leq |G|$, with a probability of at most $e^{-\Omega(k)}$, $G'[i] \neq G[i]$; and

(3) with a probability of at most $\frac{k}{n^3}$, the algorithm does not stop in $(O(k(\frac{n^2}{|G|}(\log n)^{O(1)} + h^2)), O(k))$, where n is the longest length of any input sequences, and $h = \min(|G|, n^{\frac{2}{5}})$.

Corollary 5. *There exists a randomized algorithm and positive constants, c_0, c_1 and α, such that if the alphabet size is at least four, the number of sequences is at least $c_1 \log n$, the motif length is at least $c_0 \log n$ and each character in the motif region has a probability of at most α of mutation, then the motif can be recovered with a probability of at least $\frac{3}{4}$ in $(O(\frac{n^2}{|G|}(\log n)^{O(1)}), O(\log n))$ time.*

We give a deterministic algorithm for the case with a $\Omega(1)$ mutation rate. The performance of the algorithm is stated in Theorem 6. Theorem 6 implies Corollary 7 by selecting $k = c_1 \log n$ with some constant c_1 that is large enough.

Theorem 6. *Assume that the alphabet size, t, is at least four. There exists a deterministic algorithm and a constant, c_0, such that if the length of the motif G is at least $c_0 \log n$, then, given k independent $\Theta(n, G, \mu))$-sequences, the algorithm runs in $(O(n^2(\log n)^{O(1)} + h^2 k), O(k))$ and outputs G', such that:*

(1) with a probability of at most $e^{-\Omega(k)}$, $|G'| \neq |G|$;

(2) for each $1 \leq i \leq |G|$, with a probability of at most $e^{-\Omega(k)}$, $G'[i] \neq G[i]$; and

(3) with a probability of at most $\frac{k}{n^3}$, the algorithm does not stop in $(O(k(n^2(\log n)^{O(1)} + h^2)), O(k))$ time, where n is the longest length of any input sequences, and $h = \min(|G|, n^{\frac{2}{5}})$.

Corollary 7. *There exists a deterministic algorithm and positive constants, c_0, c_1 and α, such that if the alphabet size is at least four, the number of sequences is at least $c_1 \log n$, the motif length is at least $c_0 \log n$ and each character in the motif region has a probability of at most α of mutation, then the motif can be recovered with a probability of at least $\frac{3}{4}$ in $(O(n^2(\log n)^{O(1)}), O(\log n))$ time.*

4. Algorithm Recover-Motif

In this section, we give a unified approach to describe three algorithms. The performance of the algorithms is stated in Theorems 2, 4 and 6. The description of Algorithm Recover-Motif is given in Section 4.2. The analysis of the algorithm is given at Section 6.

4.1. Some Parameters

Definition 8.

i. Parameter x is selected to be 10. This parameter controls the failure probability of our algorithms to be at most $\frac{1}{2^x}$.

ii. The size of the alphabet is $t \geq 4$.

iii. Select a constant $\rho_0 \in (0, 1)$ to have inequality (1):

$$\rho_0 < \frac{t-1}{2t} \tag{1}$$

iv. The constant $\epsilon \in (0, 1)$ is selected to satisfy:

$$\epsilon < \min((\frac{t-1}{t} - (2\rho_0 + 2\epsilon)),\ \frac{1}{5}(1 - \frac{2}{t-1} - \frac{4}{2^x}),\ \frac{1}{3}) \tag{2}$$

The existence of ϵ follows from inequality (1). The constant ϵ is used to control the mutation in the motif area. It is a part of parameter β defined in item (xiv) of this definition.

v. Let $c = e^{-\frac{\epsilon^2}{3}}$. The constant, c, is used to simplify probabilistic bounds, which are derived from the applications of Chernoff bounds (see Corollary 18).

vi. Define $r(y) = (\frac{1}{t-1} + \frac{c^y}{1-c})$.

vii. Define u_1 to be a large constant that, for all $v \geq 0$:

$$\frac{2(v+u_1)c^{v+u_1}}{(1-c)^2} \leq \frac{1}{5 \cdot 2^x} \tag{3}$$

viii. Select constant $\rho_1 \in (0, 1)$, such that:

$$\frac{2}{t-1} + \frac{4}{2^x} + 5\epsilon + \rho_1 < 1 \tag{4}$$

The existence of ρ_1 follows from $\epsilon < \frac{1}{5}(1 - \frac{2}{t-1} - \frac{4}{2^x})$, which is implied by inequality (2).

ix. Select constant $\rho_2 \in (0,1)$ and constant positive integer v that are large enough, such that:

$$\frac{6(v+u_1)c^v}{1-c} + \rho_2 < \rho_1, \quad \text{and} \tag{5}$$

$$(\frac{1}{2^x} + (v+u_1)\frac{c^v}{1-c} + \frac{c^v}{1-c} + \frac{1}{5\cdot 2^x}) \leq 1/2 \tag{6}$$

x. Define $\varsigma_0 = \frac{1}{2^x}$, and: $\varphi(v) = (v+u_1)\frac{c^v}{1-c} + \frac{c^v}{1-c})$

xi. Select constant α_0, such that:

$$4(v-1)\alpha_0 + \alpha_0 \quad < \quad \rho_2, \quad \text{and} \tag{7}$$

$$\alpha_0 \quad < \quad \rho_0 \tag{8}$$

Adding inequalities (4), (5) and (7), we have inequality (9):

$$(\frac{2}{t-1} + \frac{4}{2^x} + 5\epsilon) + \frac{6(v+u_1)c^v}{1-c} + (4(v-1)\alpha_0 + \alpha_0) < 1 \tag{9}$$

By arranging the terms in inequality (9) and the definitions of $r(v)$ and $\varphi(v)$, we have inequality (10):

$$2((2(v-1)\alpha_0 + \frac{c^v}{1-c}) + r(v) + 2(\varsigma_0 + \varphi(v)) + 2\epsilon) + (\alpha_0 + \epsilon) < 1 \tag{10}$$

xii. The maximal mutation rate, α, for the second algorithm (Theorem 4) and the third algorithm (Theorem 6) are selected as α_0. Since the mutation rate of our sublinear time algorithm is bounded by $\frac{1}{(\log n)^{2+\mu}}$, the maximal mutation rate α for the first algorithm (Theorem 2) is less than α_0 when n is large enough. We always assume that all mutation rates α in our three algorithms are in the range $(0, \alpha_0]$.

xiii. Define $q(y) = 2(v-1)\alpha + \frac{2c^y}{1-c}$. By inequality (10), the definition of $q(y)$ and the fact that $\alpha \in (0, \alpha_0)$, we have:

$$2(q(v) + r(v) + 2(\varsigma_0 + \varphi(v)) + 2\epsilon) + (\alpha_0 + \epsilon) \quad < \quad 1 \tag{11}$$

Inequality (11) implies $q(v) \leq \frac{1}{2}$. By inequality (6), we have that:

$$(\frac{1}{2^x} + (v+u_1)\frac{c^v}{1-c} + \frac{c^v}{1-c} + \frac{1}{5\cdot 2^x}) + q(v) \leq 3/4 \tag{12}$$

xiv. Let $\beta = 2\alpha + 2\epsilon$. The parameter, β, controls the similarity of $\aleph(S)$ and the original motif, G (see Lemma 27).

xv. Define $R = r(v)$.

xvi. We define the following Q_0.

$$Q_0 = q(v) \tag{13}$$

The parameter, Q_0, used in Lemma 27 gives an upper bound of the probability that a $\Theta(n, G, \alpha)$-sequence, S, whose $\aleph(S)$ will not be similar enough to the original motif, G, according to the conditions in Lemma 27.

xvii. Select constant d_0, such that:

$$n^3 c^{d_0 \log n} < \frac{1}{5 \cdot 2^x} \quad \text{for all large } n \tag{14}$$

xviii. Select constant d_1, such that $(v + u_1)c^{d_1 \log n} < \frac{1}{5 \cdot 2^x}$.

xix. Select number u_2, such that:

$$(d_1 \log n)(v + u_1)\frac{c^{v+u_2}}{1-c} \leq \frac{1}{5 \cdot 2^x}. \text{ and} \tag{15}$$

$$(v + u_1)\frac{c^{v+u_2}}{1-c} < \frac{1}{5 \cdot 2^x} \tag{16}$$

Since only n is variable, we can make $u_2 = O(\log \log n)$.

xx. For a fixed $c_0 \in (0, 1)$, define $\delta_{c_0} = \frac{\ln \frac{1}{c_0}}{2}$.

4.2. Description of Algorithm Recover-Motif

The algorithms are described in this section. The description combines three algorithms together. The simplest deterministic algorithm is also given in Section 5. Before presenting the algorithm, we define some notions.

Definition 9.

- Two sequences, X_1 and X_2, are *weakly left matched* if: (1) both $|X_1|$ and $|X_2|$ are at least $d_0 \log n$; and (2) diff$(X_1[1, i], X_2[1, i]) \leq \beta$ for all integers i, $v \leq i \leq d_0 \log n$, where v is defined in item (ix) in Definition 8.
- Two sequences, X_1 and X_2, are *left matched* if: (1) $d_0 \log n \leq |X_1|, |X_2|$; (2) $X_1[i] = X_2[i]$ for $i = 1, \cdots, v - 1$; and (3) diff$(X_1[1, i], X_2[1, i]) \leq \beta$ for all integers i, $v \leq i \leq d_0 \log n$.
- Two sequences, X_1 and X_2, are *weakly right matched* if X_1^R and X_2^R are weakly left matched, where $X^R = a_n \cdots a_1$ is the inverse sequence of $X = a_1 \cdots a_n$.
- Two sequences, X_1 and X_2, are *right matched* if X_1^R and X_2^R are left matched, where $X^R = a_n \cdots a_1$ is the inverse sequence of $X = a_1 \cdots a_n$.
- Two sequences, X_1 and X_2, are *matched* if X_1 and X_2 are both left and right matched.

Variable L will be controlled in the range $L \in [(\log n)^{3+\epsilon_1}, n^{\frac{2}{5}-\epsilon_2}]$ in our algorithm with a high probability. We define the following functions that depend on L.

Definition 10. Define $M(L) = \frac{\sqrt{3 \log n + x}}{\sqrt{1-\gamma}}\sqrt{L} \log n$. Define $M_1(L) = \frac{\delta_{c_0} M(L)}{\log n}$ (see (xx) of Definition 8 for δ_{c_0}), where $c_0 = \frac{1}{4}$.

We would like to minimize the function $(\frac{n}{L}M + L^2) \log n$. This selection can make the total time complexity sublinear.

Definition 11. For a $\Theta(n, G, \alpha)$ sequence, S, define LB(S) to be the left boundary, l, of the motif region $\aleph(S)$ in S and RB(S) to be the right boundary, r, of the motif region $\aleph(S)$ in S, such that $\aleph(S) = S[l, r]$.

4.2.1. . Boundary-Phase of Algorithm Recover-Motif

The first phase of Algorithm Recover-Motif finds the rough motif boundaries of all input sequences. It first detects the rough motif boundaries of one sequence via comparing two input sequences. Then, the rough boundaries of the first sequence are used to find the rough motif boundaries of other input sequences.

Three algorithms share most of the functions. We have a unified approach to describe them. A special variable, "algorithm-type", selects one of the three algorithms, respectively.

Definition 12. Let algorithm-type represent one of the three algorithm types, "RANDOMIZED-SUBLINEAR", "RANDOMIZED-SUBQUADRATIC" and "DETERMINISTIC-SUPERQUADRATIC".

Definition 13. Assume that A_1 is a set of positions in a $\Theta(n, G, \alpha)$ sequence, S_1, and A_2 is a set of positions in a $\Theta(n, G, \alpha)$ sequence, S_2. If there are positions $a_1 \in A_1$ and $a_2 \in A_2$, such that for some position, j, with $1 \leq j \leq |G|$, a_1 is the position of $\aleph(S_1)[j]$ in S_1 and a_2 is the position of $\aleph(S_2)[j]$ in S_2, then A_1 and A_2 have a *collision* at (a_1, a_2).

In the following function, Collision-Detection, the parameter, $\omega \leq \beta$, is defined below in the three algorithms.

$$\omega_{\text{algorithm-type}} = \begin{cases} 0 & \text{if algorithm-type=RANDOMIZED-SUBLINEAR;} \\ \beta & \text{if algorithm-type=RANDOMIZED-SUBQUADRATIC;} \\ \beta & \text{if algorithm-type=DETERMINISTIC-SUPERQUADRATIC.} \end{cases} \tag{17}$$

Function **Collision-Detection**(S_1, U_1, S_2, U_2) is used to detect a point, $a_1 \in U_1$, in the motif area in S_1 and another $a'_1 \in U_1$ point in the motif area of S_1. The two points, a_1 and a'_1, are close to the left and right motif boundaries of S_1, respectively. A similar pair of points, e_1 and e'_1, in U_2 is also derived for S_2. See the examples in Section 3.2.3. .

```
Collision-Detection$(S_1, U_1, S_2, U_2)$
Input: a pair of $\Theta(n, G, \alpha)$-sequences, $S_1$ and $S_2$; $U_i$ is a set of locations in $S_i$ for $i = 1, 2$.
Output: the left and right rough boundaries of two sequences.
    Let $D_1$ be all subsequences $S_1[a, a + d_0 \log n - 1]$ of $S_1$ of a length of $d_0 \log n$ with $a \in U_1$.
    Let $D_2$ be all subsequences $S_2[b, b + d_0 \log n - 1]$ of $S_2$ of a length of $d_0 \log n$ with $b \in U_2$.
    Find two subsequences, $X_1 = S_1[a_1, a_1 + d_0 \log n - 1] \in D_1$ and
    $X_2 = S_2[b_1, b_1 + d_0 \log n - 1] \in D_2$, such that $a_1$ is the least and $\text{diff}(X_1, X_2) \leq \omega_{\text{algorithm}-\text{type}}$.
    Find two subsequences, $X'_1 = S_1[a'_1, a'_1 + d_0 \log n - 1] \in D_1$ and
    $X'_2 = S_2[b'_1, b'_1 + d_0 \log n - 1] \in D_2$, such that $a'_1$ is the largest and
    $\text{diff}(X'_1, X'_2) \leq \omega_{\text{algorithm}-\text{type}}$.
    Find two subsequences, $Y_1 = S_1[f_1, f_1 + d_0 \log n - 1] \in D_1$ and
    $Y_2 = S_2[e_1, e_1 + d_0 \log n - 1] \in D_2$, such that $e_1$ is the least and
    $\text{diff}(Y_1, Y_2) \leq \omega_{\text{algorithm}-\text{type}}$.
    Find two subsequences, $Y'_1 = S_1[f'_1, f'_1 + d_0 \log n - 1] \in D_1$ and
```

```
        $Y'_2 = S_2[e'_1, e'_1 + d_0 \log n - 1] \in D_2$, such that $e'_1$ is the largest and
        diff$(Y'_1, Y'_2) \leq \omega_{\mathrm{algorithm-type}}$.
        Return $(a_1, a'_1, e_1, e'_1)$.
    End of Collision-Detection.
```

Definition 14. Let $[a, b]$ be an interval with two integers boundaries, a and b, and l be a positive integer parameter. Define an l-partition of $[a, b]$ to be l-$P([a, b])$, which contains the intervals $[a_1, b_1], [a_2, b_2], \cdots, [a_r, b_r]$, such that $a_1 = a, b_r = b, a_{i+1} = b_i + 1, b_i = a_i + l - 1$ for $i = 1, 2, \cdots, r - 1$ and $a_r \leq b_r \leq a_i + l - 1$.

For example, the three-partition of the interval $[1, 10]$ is 3-$P([1, 10]) = \{[1, 3], [4, 6], [7, 9], [10, 10]\}$. Function **Point-Selection**(S, L, I) will be defined differently in three different algorithms, where I is an interval of positions in sequence S and L is a positive integer parameter. For randomized algorithms, some random points are selected in L-$P(I)$. For a deterministic algorithm, all points in I are selected. See the examples in Section 3.2.2. .

```
    Point-Selection$(S, L, I)$
    Input: a pair of $\Theta(n, G, \alpha)$-sequences, $S$, a size parameter, $L$, of partition and a set of intervals, $I$, of positions in $S$.
    Output: a set, $U$, of positions from $S$, respectively.
    Steps:
    Let $U = \emptyset$.
    If algorithm-type=RANDOMIZED-SUBLINEAR or RANDOMIZED-SUBQUADRATICand
        if $(L \geq \frac{(\log n)^{3+\tau}}{100})$:
            for each interval, $I'$, in $I$, obtain its $L$-partition in $L$-$P(I')$ and for each interval, $J$, in $L$-$P(I')$,
                sample $M(L)$ (see Definition 10) random positions in $J$ and put them into $U$.
        Else,
            put every position of $I$ into $U_1$.
    If algorithm-type=DETERMINISTIC-SUPERQUADRATIC,
            put every position of $I$ into $U$.
    Return $U$.
    End of Point-Selection.
```

The function, **Improve-Boundaries**$(S_1, a_l, a_r, S_2, f_l, f_r, L)$, is used to improve the existing rough left and right boundaries, a_l and a_r, of S_1, respectively, and to improve the existing rough left and right boundaries, f_l and f_r, of S_2, respectively. We assume $a_l \in [\mathrm{LB}(S_1) - L, \mathrm{LB}(S_1) + L]$, $a_r \in [\mathrm{RB}(S_1) - L, \mathrm{LB}(S_1) + L]$, $f_l \in [\mathrm{LB}(S_2) - L, \mathrm{LB}(S_2) + L]$ and $f_r \in [\mathrm{RB}(S_2) - L, \mathrm{RB}(S_2) + L]$. After calling this function, more accurate approximate boundaries will be derived. From the probabilistic analysis, we have a good chance to get the exact motif boundaries for both S_1 and S_2. See the examples in Section 3.2.7. .

```
    Improve-Boundaries$(S_1, a_l, a_r, S_2, f_l, f_r, L)$
```

Input: a $\Theta(n, G, \alpha)$-sequence, S_1, with rough left and right boundaries, a_l and a_r, a $\Theta(n, G, \alpha)$-sequences, S_2 with rough left and right boundaries, f_l and f_r, and an approximate distance, L, to the nearest motif boundary from those rough boundaries (the parameter, L, usually has the properties of $\mathrm{LB}(S_1) \in [a_l - L, a_l]$, $\mathrm{RB}(S_1) \in [a_r, a_r + L]$, $\mathrm{LB}(S_2) \in [f_l - L, f_l]$ and $\mathrm{RB}(S_2) \in [f_r, f_r + L]$).

Output: improved rough left and right boundaries for both S_1 and S_2.

Steps:

```
Find two subsequences, $X_1 = S_1[a_1, a_1 + d_0 \log n - 1]$ and $X_2 = S_2[b_2, b_2 + d_0 \log n - 1]$, with $a_1 \in [a_l - L, a_l + L]$ and $b_2 \in [f_l - L, f_l + L]$, such that $\mathrm{diff}(X_1, X_2) \leq \beta$ and $a_1$ is the least.
Find two subsequences, $X'_1 = S_1[a'_1, a'_1 + d_0 \log n - 1]$ and $X'_2 = S_2[b'_2, b'_2 + d_0 \log n - 1]$, with $a'_1 \in [a_r - L, a_r + L]$ and $b_2 \in [f_r - L, f_r + L]$, such that $\mathrm{diff}(X'_1, X'_2) \leq \beta$ and $a'_1$ is the largest.
Find two subsequences, $Y_1 = S_1[e_1, e_1 + d_0 \log n - 1]$ and $Y_2 = S_2[f_2, f_2 + d_0 \log n - 1]$, with $e_1 \in [a_l - L, a_l + L]$ and $f_2 \in [f_l - L, f_l + L]$, such that $\mathrm{diff}(Y_1, Y_2) \leq \beta$ and $f_2$ is the least.
Find two subsequences, $Y'_1 = S_1[e'_1, e'_1 + d_0 \log n - 1]$ and $Y'_2 = S_2[f'_2, f'_2 + d_0 \log n - 1]$, with $e'_1 \in [a_r - L, a_r + L]$ and $f'_2 \in [f_r - L, f_r + L]$, such that $\mathrm{diff}(Y'_1, Y'_2) \leq \beta$ and $f'_2$ is the largest.
Return $(a_1, a'_1, f_2, f'_2)$.
```

End of Improve-Boundaries.

The function, **Initial-Boundaries**(S_1, S_2), detects the motif boundaries for two sequences, S_1 and S_2. It first detects rough motif boundaries that are controlled by parameter L. The rough boundaries will be improved to exact motif boundaries via calling **Improve-Boundaries**$(.)$. See the examples in Sections 3.2.2. –3.2.8. .

Initial-Boundaries(S_1, S_2)

Input: a pair of $\Theta(n, G, \alpha)$-sequences, S_1 and S_2.

Output: rough left boundary $\mathrm{roughLeft}_{S_1}$ of S_1, right boundary $\mathrm{roughRight}_{S_1}$ of S_1, rough left boundary $\mathrm{roughLeft}_{S_2}$ of S_2 and right boundary $\mathrm{roughRight}_{S_2}$ of S_2.

Steps:

```
Let $U_1 = U_2 = \emptyset$.
Let $L = n^{2/5}$.
Repeat.
    Let $U_1$ =Point-Selection$(S_1, L, [1, |S_1|])$.
    Let $U_2$ =Point-Selection$(S_2, L, [1, |S_2|])$.
    Let $(L_{S_1}, R_{S_1}, L_{S_2}, R_{S_2})$ =Collision-Detection$(S_1, U_1, S_2, U_2)$.
    If $(L_{S_1} \neq \emptyset$ and $R_{S_1} \neq \emptyset)$,
    then go to H.
    Else, $L = L/2$,
until $(L < \frac{1}{2}\frac{(\log n)^{3+\tau}}{100})$.
```

```
    H: Return Improve-Boundaries($(S_1, L_{S_1}, R_{S_1}, S_2, L_{S_2}, R_{S_2}, 2L)$).
End of Initial-Boundaries.
```

If L_S and R_S are the left and right motif boundaries of a sequence, S, then the motif length is $R_S - L_S + 1$. When we have the exact motif boundaries, L'_{S_i} and R'_{S_i}, for most sequences, S_i, with high probability, their motif length can be derived via the median in $\cup_i\{R'_{S_i} - L'_{S_i} + 1\}$. Therefore, we have the function, **Motif-Length-And-Boundaries(**Z_1**)**, to compute the length of the motif region.

```
Motif-Length-And-Boundaries($Z_1$)
Input: $Z_1 = \{S'_1, \cdots, S'_{2k_1}\}$ is a set of independent $\Theta(n, G, \alpha)$ sequences.
Steps:
For $i = 1$ to $k_1$,
    let $(\text{roughLeft}_{S'_{2i-1}}, \text{roughRight}_{S'_{2i}})$=Initial-Boundaries$(S'_{2i-1}, S'_{2i})$.
Let $L_1$ be the median of $\cup_{i=1}^{k_1}\{(\text{roughRight}_{S'_{2i-1}} - \text{roughLeft}_{S'_{2i-1}} + 1)\}$.
Return $L_1$.
End of Motif-Length-And-Boundaries.
```

4.2.2. . Extract-Phase of Algorithm Recover-Motif

After a set of motif candidates, W, is produced from Boundary-Phase of algorithm Recover-Motif, we use this set to match with another set of input sequences to recover the hidden motif by voting.

```
Match$(G_l, G_r, S''_i)$
Input: a motif left part, $G_l$ (which can be derived from the rough left boundary of an input sequence,
$S$), a motif right part, $G_r$, and a sequence, $S''_i$, from the group, $Z_2$, with known rough left and right
boundaries.
Output: either a rough motif region of $S''_i$ or an empty sequence, which means the failure in extracting
the motif region, $\aleph(S''_i)$, of $S''_i$.
Steps:
    Find a position, $a$, in $S''_i$ with $\text{roughLeft}_{S''_i} \le a \le \text{roughLeft}_{S''_i} + (v + u_2)$,
    such that $G_l$ and $S''_i[a, a + |G_l| - 1]$ are left matched (see Definition 9).
    Find a position, $b$, in $S''_i$ with $\text{roughRight}_{S''_i} - (v + u_2) \le b \le \text{roughRight}_{S''_i}$,
    such that $G_r$ and $S''_i[b - |G_r| + 1, b]$ are right matched (see Definition 9).
    If both $a$ and $b$ are found,
    then output $S''_i[a, b]$.
    Else, output $\emptyset$ (empty string).
End of Match.
```

If the left, G_l, and right, G_r, motif parts are known, we extract all the motif regions for all sequences in the set, Z_2, by the function, **Extract**(G_l, G_r, Z_2).

```
Extract$(G_l, G_r, Z_2)$
Input $Z_2 = \{S''_1, S''_2, \cdots, S''_{k_2}\}$ and left and right motif parts, $G_l$ and $G_r$ (see function
Match$(G_l, G_r, S_i)$).
```

```
**Steps:**
    For each $S''_i$ with $i = 1, 2, \cdots, k_2$,
        let $G''_i = \text{Match}(G_l, G_r, S''_i)$.
    Return $(G''_1, G''_2, \cdots, G''_{k_2})$.
**End of Extract.**
```

The following is Extract-Phase of algorithm Recover-Motif. It extracts the motif regions of another set, Z_2, of input sequences. The function is based on the condition that exact motif boundaries can be derived for most sequences. See the examples in Section 3.2.9. .

```
**Extract-Phase($S'$, $Z_2$):**
Input $S'$ is an input sequence with known $\text{roughLeft}_{S'}$ and $\text{roughRight}_{S'}$ for its rough left and right boundaries, respectively, and $Z_2 = \{S''_1, \cdots, S''_{k_2}\}$ is a set of input sequences.
**Steps:**
    For each subsequence $G_l = S'[a, a+d_0 \log n - 1]$ with $a \in [\text{roughLeft}_{S'}, \text{roughLeft}_{S'} + (v+u_1)]$
    and $G_r = S'[b - d_0 \log n + 1, b]$, with $b \in [\text{roughRight}_{S'} - (v + u_1), \text{roughRight}_{S'}]$,
        let $(G''_1, G''_2, \cdots, G''_{k_2})$ be the output from Extract$(G_l, G_r, Z_2)$.
        If the number of empty sequences in $G''_1, \cdots, G''_{k_2}$ is at most $(Q_0 + (R + 2\epsilon))k_2$,
        then return $(G''_1, G''_2, \cdots, G''_{k_2})$.
    Return $\emptyset$ (empty set).
**End of Extract-Phase .**
```

4.2.3. . Voting-Phase

The function, Vote$(G''_1, G''_2, \cdots, G''_{k_2})$, is to generate another sequence, G', by voting, where $G'[i]$ is the most frequent character among $G''_1[i], G''_2[i], \cdots, G''_{k_2}[i]$. See the examples in Section 3.2.10. .

```
**Voting-Phase**$(G''_1, G''_2, \cdots, G''_{k_2})$
**Input:** $\Theta(n, G, \alpha)$ sequences, $G''_1, G''_2, \cdots, G''_{k_2}$, of the same length, $m$.
**Output:** a sequence, $G'$, which is derived by voting on every position of the input sequences.
**Steps:**
    For each $j = 1, \cdots, m$,
        let $a_j$ be the most frequent character among $G''_1[j], \cdots, G''_{k_2}[j]$.
    Return $G' = a_1 \cdots a_m$.
**End of Vote.**
```

4.2.4. . Entire Algorithm Recover-Motif

The entire algorithm is described below. The input has two sets of sequences, Z_1 and Z_2. It detects the motif boundaries for the sequences in Z_1 via pairwise comparisons and, also, the motif length. The motif regions of the sequences in Z_2 are detected in the next phase and will be extracted. The original motif is recovered via voting for each column of characters among the extracted motif regions.

We maintain the sizes of Z_1 and Z_2 to be roughly equal, which implies:

$$|Z_1| = \Theta(|Z_2|) \tag{18}$$

```
Algorithm Recover-Motif (Z)
Input: $Z = Z_1 \cup Z_2$, where $Z_1 = \{S'_1, \cdots, S'_{2k_1}\}$ and $Z_2 = \{S''_1, \cdots, S''_{k_2}\}$ are two sets of input sequences.
Steps:
Preprocessing part:
For each $S \in Z_1 \cup Z_2$, let $\text{roughLeft}_S = \text{roughRight}_S = 0$ (the two boundaries are unknown).
$l_{motif}$ =MotifLengthAndBoundaries($Z_1$).
Let $L = l_{motif}/4$.
For $i = 1$ to $k_1$,
    let $U_{S'_{2i-1}}$ =Point-Selection($S'_{2i-1}, L, [\text{roughLeft}_{S'_{2i-1}} - 2L, \text{roughLeft}_{S'_{2i-1}} + 2L]$)$\cup$
                Point-Selection($S'_{2i-1}, L, [\text{roughRight}_{S'_{2i-1}} - 2L, \text{roughRight}_{S'_{2i-1}} + 2L]$).
For $j = 1$ to $k_2$,
    let $U_{S''_j}$ =Point-Selection($S''_j, L, [1, |S''_j|]$).
For $i = 1$ to $k_1$,
    for each $S''_j \in Z_2$,
        Let $(L_{S'_{2i-1}}, R_{S'_{2i-1}}, L_{S''_j}, R_{S''_j})$ =Collision-Detection($S'_{2i-1}, U_{S'_{2i-1}}, S''_j, U_{S''_j}$).
        Let $(L_{S'_{2i-1}}, R_{S'_{2i-1}}, \text{roughLeft}_{S''_j}, \text{roughRight}_{S''_j})$=
            Improve-Boundaries($S'_{2i-1}, L_{S'_{2i-1}}, R_{S'_{2i-1}}, S''_j, L_{S''_j}, R_{S''_j}, 2L$).
    Let $(G''_1, G''_2, \cdots, G''_{k_2})$ be the output from Extract-Phase($S'_{2i-1}, Z_2$).
    If $(G''_1, G''_2, \cdots, G''_{k_2})$ is not empty,
    then go to the Voting part.
Voting part:
    Return Voting-Phase($G''_1, G''_2, \cdots, G''_{k_2}$).
End of Algorithm Recover-Motif .
```

5. Deterministic Algorithm

In this section, we give a deterministic algorithm, which is a simplified version of the unified algorithm described before in Section 4.2. It is simpler than the randomized versions. The first phase of Algorithm Recover-Motif (.) finds the rough motif boundaries of all input sequences. It first detects the rough motif boundaries of one sequence via comparing two input sequences. Then, the rough boundaries of the first sequence are used to find the rough motif boundaries of other input sequences. We still let:

$$\omega_{\text{DETERMINISTIC-SUPERQUADRATIC}} = \beta. \tag{19}$$

```
Collision-Detection($S_1, S_2$)
Input: a pair of $\Theta(n, G, \alpha)$-sequences, $S_1$ and $S_2$; $U_i$ is a set of locations in $S_i$ for $i = 1, 2$.
Output: the left and right rough boundaries of two sequences.
```

```
    Let $D_1$ be all subsequences, $S_1[a, a+d_0\log n-1]$, of $S_1$ of a length of $d_0\log n$ with $a \in [1, |S_1|]$.
    Let $D_2$ be all subsequences, $S_2[b, b+d_0\log n-1]$, of $S_2$ of a length of $d_0\log n$ with $b \in [1, |S_2|]$.
    Find two subsequences, $X_1 = S_1[a_1, a_1 + d_0\log n - 1] \in D_1$ and
    $X_2 = S_2[b_1, b_1 + d_0\log n - 1] \in D_2$, such that $a_1$ is the least and $\text{diff}(X_1, X_2) \leq \omega_{\text{DETERMINISTIC-SUPERQUADRATIC}}$.
    Find two subsequences, $X_1' = S_1[a_1', a_1' + d_0\log n - 1] \in D_1$ and
    $X_2' = S_2[b_1', b_1' + d_0\log n - 1] \in D_2$, such that $a_1'$ is the largest and
    $\text{diff}(X_1', X_2') \leq \omega_{\text{DETERMINISTIC-SUPERQUADRATIC}}$.
    Find two subsequences, $Y_1 = S_1[f_1, f_1 + d_0\log n - 1] \in D_1$ and
    $Y_2 = S_2[e_1, e_1 + d_0\log n - 1] \in D_2$, such that $e_1$ is the least and
    $\text{diff}(Y_1, Y_2) \leq \omega_{\text{DETERMINISTIC-SUPERQUADRATIC}}$.
    Find two subsequences, $Y_1' = S_1[f_1', f_1' + d_0\log n - 1] \in D_1$ and
    $Y_2' = S_2[e_1', e_1' + d_0\log n - 1] \in D_2$, such that $e_1'$ is the largest and
    $\text{diff}(Y_1', Y_2') \leq \omega_{\text{DETERMINISTIC-SUPERQUADRATIC}}$.
    Return $(a_1, a_1', e_1, e_1')$.
End of Collision-Detection.
```

Function **Point-Selection**(S_1, S_2, L) is not used in the deterministic algorithm.

Improve-Boundaries$(S_1, a_l, a_r, S_2, f_l, f_r, L)$ is the same as that in the randomized algorithms.

```
Initial-Boundaries$(S_1, S_2)$
Input: a pair of $\Theta(n, G, \alpha)$-sequences, $S_1$ and $S_2$.
Output: rough left boundary $\text{roughLeft}_{S_1}$ of $S_1$, right boundary $\text{roughRight}_{S_1}$ of $S_1$, rough left boundary $\text{roughLeft}_{S_2}$ of $S_2$ and right boundary $\text{roughRight}_{S_2}$ of $S_2$.
Steps:
    Let $U_1 = U_2 = \emptyset$.
    Let $L = n^{2/5}$.
    Repeat.
        Let $(L_{S_1}, R_{S_1}, L_{S_2}, R_{S_2})$ =Collision-Detection$(S_1, S_2)$.
        If ($L_{S_1} \neq \emptyset$ and $R_{S_1} \neq \emptyset$),
        then go to H.
        Else, $L = L/2$,
    until $(L < \frac{1}{2}\frac{(\log n)^{3+\tau}}{100})$.
    H: Return Improve-Boundaries$(S_1, L_{S_l}, R_{S_1}, S_2, L_{S_2}, R_{S_2}, 2L)$.
End of Initial-Boundaries.
```

Motif-Length-And-Boundaries(Z_1**)** is the same as that before.

Match(G_l, G_r, S_i) is the same as that for the randomized algorithm.

Extract(G_l, G_r, Z_2) is the same as that for the randomized algorithm.

The following is Extract-Phase of algorithm Recover-Motif. It extracts the motif regions of another set, Z_2, of input sequences.

Extract-Phase(S', Z_2**)** is the same as that for the randomized algorithm.

Voting-Phase$(G''_1, G''_2, \cdots, G''_{k_2})$ is the same as that for the randomized algorithm.

The entire deterministic algorithm is described below. We maintain the sizes of Z_1 and Z_2 to be roughly equal.

```
Algorithm Recover-Motif (Z)
Input: $Z = Z_1 \cup Z_2$, where $Z_1 = \{S'_1, \cdots, S'_{2k_1}\}$ and $Z_2 = \{S''_1, \cdots, S''_{k_2}\}$ are two sets of input sequences.
Steps:
Preprocessing part:
For each $S \in Z_1 \cup Z_2$, let $\mathrm{roughLeft}_S = \mathrm{roughRight}_S = 0$ (the two boundaries are unknown).
$l_{motif}$ =MotifLengthAndBoundaries($Z_1$).
Let $L = l_{motif}/4$.
For $i = 1$ to $k_1$,
    for each $S''_j \in Z_2$,
        let $(L_{S'_{2i-1}}, R_{S'_{2i-1}}, L_{S''_j}, R_{S''_j})$ =Collision-Detection$(S'_{2i-1}, S''_j)$.
        Let $(L_{S'_{2i-1}}, R_{S'_{2i-1}}, \mathrm{roughLeft}_{S''_j}, \mathrm{roughRight}_{S''_j})$=
            Improve-Boundaries$(S'_{2i-1}, L_{S'_{2i-1}}, R_{S'_{2i-1}}, S''_j, L_{S''_j}, R_{S''_j}, 2L)$.
    Let $(G''_1, G''_2, \cdots, G''_{k_2})$ be the output from Extract-Phase$(S'_{2i-1}, Z_2)$.
    If $(G''_1, G''_2, \cdots, G''_{k_2})$ is not empty,
    then go to the Voting part.
Voting part:
    Return Voting-Phase$(G''_1, G''_2, \cdots, G''_{k_2})$.
End of Algorithm Recover-Motif .
```

6. Analysis of the Algorithm

The correctness of the algorithm will be proven via a series of lemmas in Sections 6.2 and 6.3. Section 6.2 is for Boundary-Phase and Section 6.3 is for Extract-Phase. Furthermore, Section 6.3 gives some lemmas for the two randomized algorithms, and Section 6.4 gives the proof for the deterministic algorithm.

6.1. Review of Some Classical Results in Probability

Some well-known results in classical probability theory are listed. The readers can skip this section if they understand them well. The inclusion of these results make the paper self-contained.

- For a list of events, $A_1, \cdots, A_m$, $\Pr[A_1 \cup A_2 \cup \cdots \cup A_m] \le \Pr[A_1] + \Pr[A_2] + \cdots + \Pr[A_m]$.
- For two independent events, A and B, $\Pr[A \cap B] = \Pr[A]\Pr[B]$.
- For a random variable, Y, $\Pr[Y \ge t] \le \frac{E[Y]}{t}$ for all positive real numbers, t. This is called Markov inequality.

The analysis of our algorithm employs the Chernoff bound [22] and Corollary 18 below, which can be derived from it (see [11]).

Theorem 15 ([22]). *Let $X_1, \cdots, X_n$ be n independent random 0–1 variables, where X_i takes one with a probability of p_i. Let $X = \sum_{i=1}^{n} X_i$, and $\mu = E[X]$. Then, for any $\delta > 0$:*

i. $\Pr(X < (1-\delta)\mu) < e^{-\frac{1}{2}\mu\delta^2}$ *and*

ii. $\Pr(X > (1+\delta)\mu) < \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu}$.

We follow the proof of Theorem 15 to make the following version of the Chernoff bound, so that it can be used in our algorithm analysis.

Theorem 16. *Let $X_1, \cdots, X_n$ be n independent random 0–1 variables, where X_i takes one with a probability of at most p. Let $X = \sum_{i=1}^{n} X_i$. Then, for any $\delta > 0$, $\Pr(X > (1+\delta)pn) < \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{pn}$.*

Define $g(\delta) = \frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}$. We note that $g(\delta)$ is always strictly less than one for all $\delta > 0$ and $g(\delta)$ is fixed if δ is a constant. This can be verified by checking that the function $f(x) = \ln \frac{e^{x}}{(1+x)^{(1+x)}} = x - (1+x)\ln(1+x)$ is decreasing and $f(0) = 0$. This is because $f'(x) = -\ln(1+x)$, which is less than zero for all $x > 0$.

Theorem 17. *Let $X_1, \cdots, X_n$ be n independent random 0–1 variables, where X_i takes one with a probability of at most p. Let $X = \sum_{i=1}^{n} X_i$. Then, for any $\delta > 0$, $\Pr(X < (1-\delta)pn) < e^{-\frac{1}{2}^{pn}\delta^2}$.*

Corollary 18 ([11]). *Let $X_1, \cdots, X_n$ be n independent random 0–1 variables, and $X = \sum_{i=1}^{n} X_i$.*
i) If X_i takes one with a probability of at most p, then for any $\frac{1}{3} > \epsilon > 0$, $\Pr(X > pn + \epsilon n) < e^{-\frac{1}{3}n\epsilon^2}$.
ii) If X_i takes one with a probability of at least p, then for any $\epsilon > 0$, $\Pr(X < pn - \epsilon n) < e^{-\frac{1}{2}n\epsilon^2}$.

6.2. Analysis of Boundary-Phase of Algorithm Recover-Motif

Lemma 19 shows that with only a small probability, a sequence can match a random sequence. It will be used to prove that when two substrings in two different $\Theta(n, G, \alpha)$-sequences are similar, they are unlikely not to coincide with the motif regions in the two $\Theta(n, G, \alpha)$-sequences, respectively.

Lemma 19. *Assume that X_1 and X_2 are two independent sequences of the same length and that every character of X_2 is a random character from Σ. Then:*

i. if $1 \le |X_1| = |X_2| < v$, then the probability that X_1 and X_2 are matched is $\le \frac{1}{t^{|X_1|}}$ ($t = ||\Sigma||$); and

ii. the probability for $\text{diff}(X_1, X_2) \le \beta$ is at most $e^{-\frac{\epsilon^2|X_1|}{3}}$.

Proof: The two statements are proven as follows.

Statement i: For every character, $X_2[j]$, with $1 \le j < v$, the probability is $\frac{1}{t}$ that $X_2[j] = X_1[j]$.

Statement ii: For every character, $X_2[j]$, with $1 \le j \le |X_2|$, the probability is $\frac{1}{t}$ for $X_2[j]$ to equal $X_1[j]$. If $\text{diff}(X_1, X_2) \le \beta$, the two sequences, X_1 and X_2, are identical in at least $(1-\beta)|X_1|$ positions, but the expected number of positions where the two sequences are identical is $\frac{1}{t}|X_1|$. The probability for $\text{diff}(X_1, X_2) \le \beta$ is at most $e^{-\frac{(1-\beta-\frac{1}{t})^2}{3}|X_1|} \le e^{-\frac{\epsilon^2}{3}|X_1|}$ by Corollary 18, Definition and 9, items (xiv) and xii, equation (8) and inequality (2) in Definition 8. ∎

Lemma 20 shows that with a small probability, an input $\Theta(n, G, \alpha)$ sequence contains a motif region that has many mutations.

Lemma 20.

i. *With a probability of at most* c^t*, a* $\Theta(n, G, \alpha)$ *sequence,* S*, changes more than* $\frac{\beta}{2}t$ *characters in its motif region,* $\aleph(S)[i, i+t-1]$*, with* $1 \le i \le i+t-1 \le |G|$*, where* c *is defined in item (v) in Definition 8.*

ii. *With a probability of at most* $\frac{c^y}{1-c}$*, a* $\Theta(n, G, \alpha)$ *sequence,* S*, changes more than* $\frac{\beta}{2}t$ *characters in its left motif region,* $\aleph(S)[1, t]$*, for some* t *with* $y \le t \le |G|$*, where* c *is defined in item (v) in Definition 8.*

Proof: Statement i: Every character in the $\aleph(S)$ region has a probability of at most α to mutate. We know that $|\aleph(S)| = |G| \ge d$. By Corollary 18, with a probability of at most $e^{-\frac{\epsilon^2}{3}t}$, $\aleph(S)[i, i+t-1]$ has more than $(\alpha + \epsilon)t$ mutations.

Statement ii: We know that $|\aleph(S)| = |G| \ge d$. By Corollary 18, with a probability of at most $e^{-\frac{\epsilon^2}{3}t}$, a sequence, S, in Z_1 has more than $(\alpha + \epsilon)t$ mutations (recall the setting for β in Definition 9) among the first left t characters. The total probability is at most $\sum_{t=y}^{\infty} e^{-\frac{\epsilon^2}{3}t} = \frac{c^y}{1-c}$. ▮

Lemma 21 shows that Improve-Boundaries() has a good chance to improve the accuracy of rough motif boundaries. Note that $\mathrm{LB}(S)$ and $\mathrm{RB}(S)$ are the left and right motif boundaries of S, respectively (see Definition 11).

Lemma 21. *Assume that* $\Theta(n, G, \alpha)$ *sequence* S_i *has* $L_{S_i} \in [\mathrm{LB}(S_i) - L, \mathrm{LB}(S_i) + L]$ *and* $R_{S_i} \in [\mathrm{RB}(S_i) - L, \mathrm{RB}(S_i) + L]$ *for* $i = 1, 2$*. Then, for* $(\mathrm{roughLeft}_{S_1}, \mathrm{roughRight}_{S_1}, \mathrm{roughLeft}_{S_2}, \mathrm{roughRight}_{S_2})$*=Improve-Boundaries*$(S_1, L_{S_1}, R_{S_1}, S_2, L_{S_2}, R_{S_2}, L)$*, we have the following two facts:*

i. *with a probability of at most* $\frac{2c^v}{1-c} + \frac{2(v+u)c^{v+u}}{(1-c)^2} + \frac{1}{5 \cdot 2^x n}$*;* $\mathrm{roughLeft}_{S_i}$ *is not in* $[\mathrm{LB}(S_i) - (v+u), \mathrm{LB}(S_i)]$ *for* $i = 1, 2$*.*

ii. *with a probability of at most* $\frac{2c^v}{1-c} + \frac{2(v+u)c^{v+u}}{(1-c)^2} + \frac{1}{5 \cdot 2^x n}$*,* $\mathrm{roughRight}_{S_i}$ *is not in* $[\mathrm{RB}(S_i), \mathrm{RB}(S_i) + (v+u)]$ *for* $i = 1, 2$*.*

iii. *Improve-Boundaries*$(S_1, L_{S_1}, R_{S_1}, S_2, L_{S_2}, R_{S_2}, L)$ *runs in* $O(L^2 \log n)$ *time.*

Proof: We need a bound for the following inequality:

$$\sum_{i=j}^{\infty} ia^i < \frac{ja^j}{(1-a)^2} \tag{20}$$

Let $f(x) = \sum_{i=j}^{\infty} e^{\theta ix}$. Compute the derivative $f'(x) = \theta \sum_{i=j}^{\infty} ie^{\theta ix}$. We also have the closed form for the function $f(x) = \frac{e^{\theta jx}}{1-e^{\theta x}}$, which implies:

$$\begin{aligned} f'(x) &= \frac{\theta je^{\theta jx}(1-e^{\theta x}) - e^{\theta jx}(-\theta e^{\theta x})}{(1-e^{\theta x})^2} && (21) \\ &= \frac{\theta je^{\theta jx} - \theta(j-1)e^{\theta(j+1)x}}{(1-e^{\theta x})^2} && (22) \end{aligned}$$

Let $\theta = \ln a$ and $x = 1$. We have $\sum_{i=j}^{\infty} ia^i = \frac{ja^j - (j-1)a^{j+1}}{(1-a)^2} < \frac{ja^j}{(1-a)^2}$.

Statement i. By Lemma 20, with a probability of at most $\frac{2c^v}{1-c}$, one of the left motif first y character region of S_i will change $\frac{\beta}{2y}$ characters. Therefore, with a probability of at most $P_1 = \frac{2c^v}{1-c}$, $\text{roughLeft}_{S_i} > \text{LB}(S_i)$.

For a pair of positions, p, in S_1, and q, in S_2, without loss of generality, assume that p has a larger distance to the left boundary $\text{LB}(S_1)$ of S_1 than q to the left boundary $\text{LB}(S_2)$ of S_2. Let $v + y$ be the distance from p to the left boundary $\text{LB}(S_1)$ of S_1.

By Lemma 19, the probability is at most c^{v+y} that there will be a match. There are at most $(v + y)$ cases for q. With a probability of at most $P_2 = 2\sum_{y=u}^{\infty}(v + y)c^{v+y} < \frac{2(v+u)c^{v+u}}{(1-c)^2}$, by inequality (20), $\text{roughLeft}_{S_1} < LB(S_1) - (v + u)$.

For the cases that one position is in a random region and has a distance more than $d_0 \log n$ from the left boundary, the probability is at most $P_3 = n^2 c^{d_0 \log n} < \frac{1}{5 \cdot 2^x n}$ by inequality (14).

Therefore, we have a total probability of at most $P_1 + P_2 + P_3$ that roughLeft_{S_1} is not in $[\text{LB}(S_1) - (v + u), \text{LB}(S_1)]$.

Statement ii. One can also provide a symmetric analogous proof for this statement.

Statement iii. The computation time easily follows from the implementation of Improve-Boundaries $(S_1, L_{S_1}, R_{S_1}, S_2, L_{S_2}, R_{S_2})$. ∎

Lemma 22. *Assume that for each L with $0 < L \leq \frac{|G|}{2}$, with a probability of at most $\varsigma(n)$, $L_{S_i} \notin [\text{LB}_{S_i} - L$,
$\text{LB}_{S_i} + L]$ for $i = 1, 2$, where $(L_{S_1}, R_{S_1}, L_{S_2}, R_{S_2})$ = Collision-Detection(S_1, U_1, S_2, U_2), U_1 = Point-Selection(S_1, L) and U_2 = Point-Selection(S_2, L). Then, with a probability of at most $\varsigma(n) + \frac{2(v+u_1)c^{v+u_1}}{(1-c)^2} + \frac{c^v}{1-c} + \frac{1}{5 \cdot 2^x n}$, Initial-Boundary$(S_1, S_2)$ returns $(L_{S_1}, R_{S_1}, L_{S_2}, R_{S_2})$ with $L_{S_i} \notin [\text{LB}(S_i) - (v + u_1), \text{LB}(S_i)]$ or $R_{S_i} \notin [\text{RB}(S_i), \text{RB}(S_i) + (v + u_1))]$ for $i = 1, 2$.*

Proof: It follows from Lemma 21: ∎

Lemma 23. *Assume that with a probability of $p < 0.5$, each S'_{2i-1} has its rough boundaries, $\text{roughLeft}_{S'_{2i-1}} \notin [\text{LB}(S'_{2i-1}) - u, \text{LB}(S'_{2i-1})]$ or $\text{roughRight}_{S'_{2i-1}} \notin [\text{RB}(S'_{2i-1}), \text{RB}(S'_{2i-1}) + u]$; then, with a probability of at most $e^{-(0.5-p-\epsilon)^2 k_1/3}$, l_{motif} is not in $[|G| - 2u, |G| + 2u]$, where l_{motif} is selected as the median of $\cup_{i=1}^{k_1}\{(\text{roughRight}_{S'_{2i-1}} - \text{roughLeft}_{S'_{2i-1}})\}$.*

Proof: If both $\text{roughLeft}_{S'_{2i-1}} \in [\text{LB}(S'_{2i-1}) - u, \text{LB}(S'_{2i-1})]$ and $\text{roughRight}_{S'_{2i-1}} \in [\text{RB}(S'_{2i-1}), \text{RB}(S'_{2i-1}) + u]$, then $(\text{roughRight}_{S'_{2i-1}} - \text{roughLeft}_{S'_{2i-1}})$ is in $[|G| - 2u, |G| + 2u]$.

If the median of $\cup_{i=1}^{k_1}\{(\text{roughRight}_{S'_{2i-1}} - \text{roughLeft}_{S'_{2i-1}})\}$ is not in $[|G| - 2u, |G| + 2u]$, then there are at least $\lfloor k_1 \rfloor$ numbers, i, to have $\text{roughLeft}_{S'_{2i-1}} \notin [\text{LB}(S'_{2i-1}) - u, \text{LB}(S'_{2i-1})]$ or $\text{roughRight}_{S'_{2i-1}} \notin [\text{RB}(S'_{2i-1}), \text{RB}(S'_{2i-1}) + u]$.

On the other hand, the probability is at most p, $\text{roughLeft}_{S'_{2i-1}} \notin [\text{LB}(S'_{2i-1}) - u, \text{LB}(S'_{2i-1})]$ or $\text{roughRight}_{S'_{2i-1}} \notin [\text{RB}(S'_{2i-1}), \text{RB}(S'_{2i-1}) + u]$. Therefore, this lemma follows from Corollary 18. ∎

For a $\Theta(n, G, \alpha)$-sequence S, we often obtain its left rough boundary with $\text{roughLeft}_S \leq \text{LB}(S)$. Sometimes, its exact left boundary may be missed in the algorithm.

Definition 24.

- A $\Theta(n, G, \alpha)$-sequence, S, misses its left boundary if $\mathrm{roughLeft}_S > \mathrm{LB}(S)$.
- A $\Theta(n, G, \alpha)$-sequence, S, misses its right boundary if $\mathrm{roughRight}_S < \mathrm{RB}(S)$.

Definition 25.

i. A $\Theta(n, G, \alpha)$-sequence, S, contains a *left half stable* motif region, $\aleph(S)$, if $\mathrm{diff}(G'[1, h], G[1, h]) \le \frac{\beta}{2}$ for all $h = v, v + 1, \cdots, m$, where $G' = \aleph(S)$, and $m = |G|$, as defined in Definition 8 and Section 2, respectively.

ii. A $\Theta(n, G, \alpha)$-sequence, S, contains a *right half stable* motif region, $\aleph(S)$, if $\mathrm{diff}(G'[m - h, m], G[m - h, m]) \le \frac{\beta}{2}$ for $h = v - 1, v + 1, \cdots, m - 1$, where $G' = \aleph(S)$ and $m = |G|$.

iii. A $\Theta(n, G, \alpha)$-sequence, S, contains a *stable* motif region, $\aleph(S)$, satisfying the following conditions: (1) $G'[i] = G[i]$ for $i = 1, \cdots, v-1$; (2) $G'[m-i+1] = G[m-i+1]$ for $i = 1, \cdots, v-1$; and (3) the S motif region is both left and right half stable, where $G' = \aleph(S)$ and $m = |G|$.

Lemma 26. *Assume that:*

- $l_{motif} \in [|G| - 2(v + u_1), |G| + 2(v + u_1)]$;
- *S contains both a left half and a right half stable motif region and* $\mathrm{roughLeft}_S \in [\mathrm{LB}(S) - (v + u_1), \mathrm{LB}(S)]$ *and* $\mathrm{roughRight}_S \in [\mathrm{RB}(S), \mathrm{RB}(S) + (v + u_1)]$ *(see Definition 8 for u_1 and v); and*
- *for each L with $(v + u_1) < L \le \frac{|G|}{2}$, if S_1 has* $\mathrm{roughLeft}_{S_1} \in [\mathrm{LB}_{S_1} - L, \mathrm{LB}_{S_1} + L]$ *and* $\mathrm{roughRight}_{S_1} \in [\mathrm{RB}_{S_1} - L, \mathrm{RB}_{S_1} + L]$, *then with a probability of at most $\varsigma(n)$, $L_{S''_i} \notin [\mathrm{LB}_{S''_i} - 2L, \mathrm{LB}_{S''_i} + 2L]$ for $i = 1, 2$, where $(L_{S_1}, R_{S_1}, L_{S''_i}, R_{S''_i}) =$ Collision-Detection (S_1, U_1, S''_i, U_2), $U_1 =$ Point-Selection$(S_1, L, [\mathrm{roughLeft}_{S_1} - 2L, \mathrm{roughLeft}_{S_1} + 2L]) \cup$ Point-Selection$(S_1, L, [\mathrm{roughRight}_{S_1} - 2L, \mathrm{roughRight}_{S_1} + 2L])$ and $U_2 =$ Point-Selection $(S''_i, L, [1, |S''_i|])$.*
- *The rough boundaries for all sequences, $S''_i \in Z_2$, are computed via $(L_S, R_S, L_{S''_i}, R_{S''_i}) =$ Collision-Detection$(S, U_S, S''_i, U_{S''_i})$ and* $(L_S, R_S, \mathrm{roughLeft}_{S''_i}, \mathrm{roughRight}_{S''_i}) =$ *Improve-Boundaries$(S, L_S, R_S, S''_i, L_{S''_i}, R_{S''_i}, 2L)$.*

Then, with a probability of at most $e^{-\frac{\epsilon^2 k_2}{3}}$, there are more than $(2(\varsigma(n) + (v + u_1)\frac{c^{v+u}}{1-c} + \frac{c^v}{1-c}) + \epsilon)k_2$ sequences S''_i in $\{S''_1, \cdots, S''_{k_2}\}$ with $\mathrm{roughLeft}(S''_i) \notin [\mathrm{LB}(S''_i) - (v+u), \mathrm{LB}(S''_i)]$ *or* $\mathrm{roughRight}(S''_i) \notin [\mathrm{RB}(S''_i), \mathrm{RB}(S''_i) + (v + u)]$.

Proof: According to the condition of this lemma, with a probability of at most $P_1 = \varsigma(n)$, $L_{S''_i} \notin [\mathrm{LB}_{S''_i} - 2L, \mathrm{LB}_{S''_i} + 2L]$, where $(L_S, R_S, L_{S''_i}, R_{S''_i}) =$ Collision-Detection(S, U_1, S''_i, U_2) and $(U_1, U_2) =$ Point-Selection(S, S''_i, L).

For a fixed pattern from S, by Lemma 19, with a probability of at most $\sum_{y=v+u}^{\infty} c^y = \frac{c^{v+u}}{1-c}$, it has a distance of more than $v + u$ to the true left boundary. As we need to deal with $v + u_1$ possible patterns from S, with a probability of at most $P_{2,l} = (v + u_1) \cdot \frac{c^{v+u}}{1-c}$, $\mathrm{roughLeft}_{S''_i} < \mathrm{LB}(S''_i) - (v + u)$.

Similarly, with a probability of at most $P_{2,r} = (v + u_1)\frac{c^{v+u}}{1-c}$, $\text{roughRight}_{S_i''} > \text{RB}(S_i'') + (v + u)$. Let $P_2 = P_{2,l} + P_{2,r}$.

With a probability of at most $P_{3,l} = \frac{c^v}{1-c}$, S_i'' does not contain a left half stable motif region by Lemma 20. Similarly, with a probability of at most $P_{3,r} = \frac{c^v}{1-c}$, S_i'' does not contain a right half stable motif region. Let $P_3 = P_{3,l} + P_{3,r}$.

Although, S is involved in searching the left boundary with all other sequences. The non-missing condition is to let each sequence not change too many characters in the motif region. Therefore, this is an independent event for each sequence. It is safe to use the Chernoff bound to deal with it.

With a probability of at most $P = e^{-\frac{\epsilon^2 k_2}{3}}$, there are more than $(P_1 + P_2 + P_3 + \epsilon)k_2$ sequences, S_i'', in $\{S_1'', \cdots, S_{k_2}''\}$ with $\text{roughLeft}(S_i'') \notin [\text{LB}(S_i'') - (v + u), \text{LB}(S_i'')]$ or $\text{roughRight}(S_i'') \notin [\text{RB}(S_i''), \text{RB}(S_i'') + (v + u)]$. ▮

6.3. Analysis of Extract-Phase and Voting-Phase of Algorithm Recover-Motif

Lemma 27 shows that with a high probability, the left and last parts of the motif region in a $\Theta(n, G, \alpha)$-sequence do not change much.

Lemma 27. *With a probability of at most Q_0 defined in Equation (13), a $\Theta(n, G, \alpha)$-sequence, S, does not contain a stable motif region.*

Proof: The probability is $V_1 = 2(v - 1)\alpha$ not to satisfy conditions (1) and (2) of item (iii) in Definition 25. Consider condition (3) of item (iii) in Definition 25. Since every character of $\aleph(S)[1, m]$ (notice that $m = |G|$) has a probability of at most α to mutate, by Corollary 18, the probability is at most $e^{-\frac{1}{3}\epsilon^2 r}$ that $\text{diff}(G[1, h], G'[1, h]) > \frac{\beta}{2} = \alpha + \epsilon$. Let $V_3 = \sum_{r=v}^{\infty} e^{-\frac{1}{3}\epsilon^2 r} = \frac{c^v}{1-c}$, where $c = e^{-\frac{1}{3}\epsilon^2}$, as defined in Definition 8. Therefore, the probability is at most V_3 that $\text{diff}(G[1, h], G'[1, h]) > \frac{\beta}{2} = \alpha + \epsilon$ for some $h \in \{v, v + 1, \cdots, m\}$. Similarly, we define $V_4 = \sum_{r=v}^{\infty} e^{-\frac{1}{3}\epsilon^2 r} \leq \frac{c^v}{1-c}$ for the probability on the right-hand side. The probability is at most V_4 that $\text{diff}(G[m - h, m], G'[m - h, m]) > \frac{\beta}{2} = \alpha + \epsilon$ for some $h \in \{v, v + 1, \cdots, m\}$. The probability that S does not contain a stable motif region is at most $V_1 + V_3 + V_4 = Q_0$. ▮

Definition 28. Assume that $Z_1 = \{S_1', \cdots, S_{2k_1}'\}$ contains S_{2i-1}', which contains a stable motif region. We fix such a S_{2i-1}'.

- Define $G_L = \aleph(S_{2i-1}')[1, d_0 \log n - 1]$ to be the left part of the motif region, $\aleph(S_{2i-1}')$.
- Define $G_R = \aleph(S_{2i-1}')[|G| - (d_0 \log n) + 1, |G|]$ to be the right part of the motif region, $\aleph(S_{2i-1}')$.

Lemma 29 shows that with a high probability, Extract-Phase of algorithm Recover-Motif extracts the correct motif regions from the sequences in Z_1. It uses G'' to match $\aleph(S)$ in another sequences, S. The parameter, R, gives a small probability that the matched region between G'' and S is not in $\aleph(S)$.

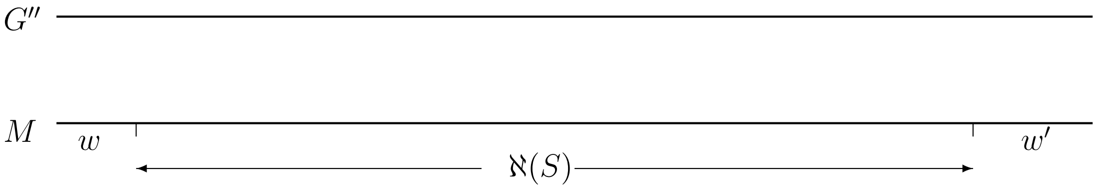
Lemma 29.

i. *Assume that G_l and G_r are fixed sequences of length $d_0 \log n$. Let S be a $\Theta(n, G, \alpha)$-sequence with $M = \mathrm{Match}(G_l, G_r, S)$, and let w_0 be the number of characters of M that are not in the region of $\aleph(S)$. Then, the probability is at most R that $w_0 \geq 1$, where R is defined in (xv) of Definition 8.*

ii. *The probability is at most Q_0 that given a $\Theta(n, G, \alpha)$-sequence S, $\mathrm{Match}(G_L, G_R, S) = \emptyset$.*

Proof: Assume that $w_0 \geq 1$. Let w be the number of characters outside of $\aleph(S)$ on the left of M, and let w' be the number of characters outside of $\aleph(S)$ on the right of M. Clearly, $w_0 = w + w'$. Since $w_0 \geq 1$, either $w \geq 1$ or $w' \geq 1$. See Figure 1. Without loss of generality, we assume $w \geq 1$.

Figure 1. G'' and M.



Statement i: There are two cases.

Case (a): $1 \leq w < v$. By Lemma 19, the probability for this case is at most $\frac{1}{t}$ for a fixed w. The total probability for this case for $1 \leq w < v$ is at most $\sum_{i=1}^{v-1} \frac{1}{t^i} \leq \sum_{i=1}^{\infty} \frac{1}{t^i} = \frac{1}{t-1}$.

Case (b): $v \leq w$. By Lemma 19, the probability is at most $e^{-\frac{\epsilon^2}{3}w}$ for a fixed w. The total probability for $v \leq w$ is at most $\sum_{w=v}^{\infty} e^{-\frac{\epsilon^2}{3}w} = \frac{c^v}{1-c}$.

The probability analysis is similar when $w' \geq 1$. Therefore, the probability for this case is at most $R = (\frac{1}{t-1} + \frac{c^v}{1-c})$ for $w_0 \geq 1$.

Statement ii: By Lemma 27, with a probability of at most Q_0, S does not contain a stable motif region. Therefore, we have a probability of at most Q_0 that given a random $\Theta(n, G, \alpha)$-sequence, S, $\mathrm{Match}(G_L, G_R, S) = \emptyset$. ∎

Lemma 30 shows that we can use G_l and G_r to extract most of the motif regions for the sequences in Z_2 if $G' = G_L$ (recall that G_L is defined right after Lemma 27).

Lemma 30. *Assume that G_l and G_r are two sequences of a length of $d_0 \log n$, and $G_i = \mathrm{Match}(G_l, G_r, S_i'')$ for $S_i'' \in Z_2 = \{S_1'', \cdots, S_{k_2}''\}$ and $i = 1, \cdots, k_2$ (recall that each sequence, G_i, is either an empty sequence or a sequence of the length of $|G_l|$).*

i. *If $G_l = G_L$, $G_r = G_R$ and there are no more than yk_2 ($y \in [0, 1]$) sequences, S_i'', with $\mathrm{roughLeft}_{S_i''} \notin [\mathrm{LB}(S_i'') - (v + u_2), \mathrm{LB}(S_i'')]$ or $\mathrm{roughRight}_{S_i''} \notin [\mathrm{RB}(S_i''), \mathrm{RB}(S_i'') + (v + u_2)]$, then the probability is at most $e^{-\frac{\epsilon^2 k_2}{3}}$ that there are more than $(Q_0 + y + \epsilon)k_2$ sequences, G_i, with $G_i = \emptyset$.*

ii. *For arbitrary G_l and G_r, with a probability of at most $e^{-\frac{\epsilon^2 k_2}{3}}$, $|\{i | G_i \neq \emptyset \text{ and } G_i \neq \aleph(S_i''), i = 1, \cdots, k_2\}| > (R + \epsilon)k_2$, where R is defined in Definition 8.*

Proof: Recall that sequence G_{1L} is selected right after Lemma 27.

Statement i: By Lemma 29, for every $S_i'' \in Z_2$, the probability is at most Q_0 that S_i'' does not contain a stable motif region, $\aleph(S_i'')$. By Corollary 18, we have a probability of at most $e^{-\frac{\epsilon^2 k_2}{3}}$ that there are more than $(Q_0 + y + \epsilon)k_2$ sequences, G_i, with $G_i = \emptyset$.

Statement ii: By Lemma 29, the probability is at most R that $G_i \neq \aleph(S_i'')$. By Corollary 18, with a probability of at most $e^{-\frac{\epsilon^2 k_2}{3}}$, $|\{i | G_i \neq \aleph(S_i''), i = 1, \cdots, k_2\}| > (R + \epsilon)k_2$. ∎

Definition 31.

- Given two sequences, G_r and G_r, define:

  $M(G_r, G_r) = \{G_i'' : G_i'' =$Match$(G_l, G_r, \mathrm{roughLeft}_{S_i''}, \mathrm{roughRight}_{S_i''}, S_i'')\ i = 1, \cdots, k_2\}$.

- For a $\Theta(n, G, \alpha)$ sequence, S, define $G_{S,L}$ to be the $\aleph(S)[1, d_0 \log n]$, which is the leftmost subsequence of a length of $d_0 \log n$ in the motif region of S.

- For a $\Theta(n, G, \alpha)$ sequence, S, define $G_{S,R}$ to be the $\aleph(S)[m - d_0 \log n + 1, m]$, which is the rightmost subsequence of a length of $d_0 \log n$ in the motif region of S, where $m = |G| = |\aleph(S)|$;

the condition iv of Lemma 32.

Lemma 32. *Assume that we have the following conditions:*

i. For each L with $0 < L \leq \frac{|G|}{2}$, with a probability of at most $\varsigma_1(n)$, $L_{S_i} \notin [\mathrm{LB}_{S_i} - 2L, \mathrm{LB}_{S_i} + 2L]$ or $R_{S_i} \notin [\mathrm{RB}_{S_i} - 2L, \mathrm{RB}_{S_i} + 2L]$ for $i = 1, 2$, where $(L_{S_1}, R_{S_1}, L_{S_2}, R_{S_2})$ =Collision-Detection(S_1, U_1, S_2, U_2), U_1 =Point-Selection$(S_1, L, [1, |S_1|])$ and U_2 =Point-Selection$(S_2, L, [1, |S_2|])$.

ii. For each L with $0 < L \leq \frac{|G|}{2}$, if S_1 has $\mathrm{roughLeft}_{S_1} \in [\mathrm{LB}_{S_1} - L, \mathrm{LB}_{S_1} + L]$ and $\mathrm{roughRight}_{S_1} \in [\mathrm{RB}_{S_1} - L, \mathrm{RB}_{S_1} + L]$, then with a probability of at most $\varsigma_2(n)$, $L_{S_i''} \notin [\mathrm{LB}_{S_i''} - 2L, \mathrm{LB}_{S_i''} + 2L]$ for $i = 1$ or $i = 2$, where $(L_{S_1}, R_{S_1}, L_{S_i''}, R_{S_i''})$ = Collision-Detection(S_1, U_1, S_i'', U_2), U_1 = Point-Selection$(S_1, L, [\mathrm{roughLeft}_{S_1} - 2L, \mathrm{roughLeft}_{S_1} + 2L])\cup$ Point-Selection $(S_1, L, [\mathrm{roughRight}_{S_1} - 2L, \mathrm{roughRight}_{S_1} + 2L])$ and $U_2 =$ Point-Selection $(S_i'', L, [1, |S_i''|])$.

iii. The inequality $(P_0 + Q_0) < c_0$ holds for some constant $c_0 < 1$, where Q_0 is defined in Equation (13) and $P_0 = \varsigma_1(n) + \frac{2(v+u_1)c^{v+u_1}}{(1-c)^2} + \frac{c^v}{1-c} + \frac{1}{5 \cdot 2^x n}$.

iv. The inequality $1 - 2(Q_0 + V_0 + (R + 2\epsilon)) - (\alpha + \epsilon) > 0$ holds, where $V_0 = (2(\varsigma_2(n) + (v + u_1)\frac{c^{v+u_2}}{1-c} + \frac{c^v}{1-c}) + \epsilon)$.

Then, the algorithm generates a set of at most k_2 subsequences for voting and votes for a sequence, G', such that:

(1) with a probability of at most $e^{-\Omega(k_1)} + e^{-\Omega(k_2)}$, $|G'| \neq |G|$; and

(2) for each $1 \leq i \leq |G|$, with a probability of at most $e^{-\Omega(k_1)} + e^{-\Omega(k_2)}$, $G'[i] \neq G[i]$.

Before proving Lemma 32, we note that both $\varsigma_1(n)$ and $\varsigma_2(n)$ are at most $\frac{1}{2^x n^3}$ for all of the three algorithms. They will be proven by Lemma 47 and Lemma 46 for the case algorithm-type=RANDOMIZED-SUBLINEAR, Lemma 41 and Lemma 40 for the case algorithm-type=RANDOMIZED-SUBQUADRATIC, and Lemma 35 for the case algorithm-type=DETERMINISTIC-SUPERQUADRATIC.

Proof:

By Lemmas 22, with a probability of at most $P_0 = \varsigma_1(n) + \frac{2(v+u_1)c^{v+u_1}}{(1-c)^2} + \frac{c^v}{1-c} + \frac{1}{5 \cdot 2^x n}$, $\text{roughLeft}_{S'_{2i-1}} \notin [\text{LB}(S'_{2i-1}) - (v+u_1), \text{LB}(S'_{2i-1})]$ or $\text{roughRight}_{S'_{2i-1}} \notin [\text{RB}(S'_{2i-1}), \text{RB}(S'_{2i-1}) + (v+u_1)]$.

By Lemma 23, with a probability of at most $P_a = e^{-(0.5-P_0-\epsilon)^2 k_1/3} = e^{-\Omega(k_1)}$, the approximate motif length, l_{motif}, is not in the range $[|G| - 2(v+u_1), |G| + 2(v+u_1)]$. Assume $l_{motif} \in [|G| - 2(v+u_1), |G| + 2(v+u_1)]$ in the rest of the proof of this lemma.

By Lemma 27, with a probability of at most Q_0, a $\Theta(n, G, \alpha)$ sequence does not contain a stable motif region. Therefore, with a probability of at most $P_1 = (P_0 + Q_0)^{k_1}$, the following statement is false:

(i) One of S'_{2i-1} for $i = 1, \cdots, k_1$ has $\text{roughLeft}_{S'_{2i-1}} \in [\text{LB}(S'_{2i-1}) - (v+u_1), \text{LB}(S'_{2i-1})]$, $\text{roughRight}_{S'_{2i-1}} \in [\text{RB}(S'_{2i-1}), \text{RB}(S'_{2i-1}) + (v+u_1)]$ and has a stable motif region.

By Lemma 26, with a probability of at most $P_2 = e^{-\frac{\epsilon^2 k_2}{3}}$, there are more than $(2(\varsigma_2(n) + (v+u_1)\frac{c^{v+u_2}}{1-c} + \frac{c^v}{1-c}) + \epsilon)k_2$ sequences S''_i with $\text{roughLeft}_{S''_i} \notin [\text{LB}(S''_i) - (v+u_2), \text{LB}(S''_i)]$ or $\text{roughRight}_{S''_i} \notin [\text{RB}(S''_i), \text{LB}(S''_i) + (v+u_2)]$. In other words, with a probability of at most P_2, the following statement is false:

(ii) There are no more than $V_0 k_2$ sequences S''_i with $\text{roughLeft}_{S''_i} \notin [\text{LB}(S''_i) - (v+u_2), \text{LB}(S''_i)]$ or $\text{roughRight}_{S''_i} \notin [\text{RB}(S''_i), \text{RB}(S''_i) + (v+u_2)]$, where $V_0 = (2(\varsigma_2(n) + (v+u_1)\frac{c^{v+u_2}}{1-c} + \frac{c^v}{1-c}) + \epsilon)$.

Assume that Statement (ii) is true. By Lemma 30, with a probability of at most $P_3 = c^{k_2}$, the following statement is false:

(iii) $M(G_L, G_R)$ contains at most $(Q_0 + V_0 + \epsilon)k_2$ empty sequences.

We start from the rough left boundary, roughLeft_1, of S_1 to match the other left boundaries of S''_i for $i = 1, \cdots, k_2$. There are in total at most $2(v+u_1)$ candidates to consider.

By Lemma 30, if $M(G_l, G_r)$, which consists of k_2 matched regions, has at most $(Q_0 + V_0 + \epsilon)k_2$ empty sequences, then it has more than $(R+\epsilon)k_2$ from non-motif regions with a probability of at most $P_4 = 2(v+u_1)e^{-\frac{\epsilon^2 k_2}{3}}$. After the pattern is fixed, those events in the matching are considered to be independent of each other. This is why we can apply the Chernoff bound to deal with them. Therefore, the probability is at most P_4; the following statement is false:

(iv) If $M(G_l, G_r)$ contains at most $(Q_0+V_0+\epsilon)k_2$ empty sequences, then $M(G_l, G_r)$ contains at most $(Q_0+V_0+\epsilon+(R+\epsilon))k_2 = (Q_0+V_0+(R+2\epsilon))k_2$ elements not from motif regions $\{\aleph(S''_i) : 1 \leq i \leq k_2\}$.

Therefore, with a probability of at most $P_a + P_1 + P_2 + P_3 + P_4 = e^{-\Omega(k_1)} + e^{-\Omega(k_2)}$, the sequences are not ready for voting in the next phase, which means the following two conditions are satisfied:

(a) There exists G_l and G_r generated by the algorithm, such that $M(G_l, G_r)$ contains at most $(Q_0 + V_0 + (R+2\epsilon))k_2$ elements not from motif regions $\{\aleph(S''_i) : 1 \leq i \leq k_2\}$.

(b) For every G_l and G_r that $M(G_l, G_r)$ contains at most $(Q_0 + V_0 + \epsilon)k_2$ empty sequences generated by the algorithm, $M(G_l, G_r)$ contains at most $(Q_0 + V_0 + \epsilon + (R+\epsilon))k_2 = (Q_0 + V_0 + (R+2\epsilon))k_2$ elements not from motif regions $\{\aleph(S''_i) : 1 \leq i \leq k_2\}$.

Statement (1): For a $M(G_l, G_r)$ with at most $(Q_0+V_0+(R+2\epsilon))k_2$ elements not from motif regions $\{\aleph(S''_i) : 1 \le i \le k_2\}$, we still have $k_2-(Q_0+V_0+(R+2\epsilon))k_2$ elements in $M(G_l, G_r)$ from motif regions $\{\aleph(S''_i) : 1 \le i \le k_2\}$. By the condition (iv) in this lemma, we have $k_2-(Q_0+V_0+(R+2\epsilon))k_2 > (Q_0+V_0+(R+2\epsilon))k_2$. Therefore, $|G'|$ is selected to be the length of G in the Voting-Phase().

Statement (2): For a $M(G_l, G_r) = \{G''_1, \cdots, G''_{k_2}\}$ with at most $(Q_0+V_0+(R+2\epsilon))k_2$ elements not from motif regions $\{\aleph(S''_i) : 1 \le i \le k_2\}$, we still have $k_2-(Q_0+V_0+(R+2\epsilon))k_2$ elements in $M(G_l, G_r)$ from motif regions $\{\aleph(S''_i) : 1 \le i \le k_2\}$. By Corollary 18, with a probability of at most $e^{-\frac{\epsilon^2 k_2}{3}}$, there are more than $(\alpha+\epsilon)k_2$ characters that are mutated in the same position among all k_2 motif regions for the sequences in Z_2. We have that $k_2-(Q_0+V_0+(R+2\epsilon))k_2-(\alpha+\epsilon)k_2 > (Q_0+V_0+(R+2\epsilon))k_2$ by the condition (iv) in this lemma. We let $G'[j]$ be the most frequent character among $G''_1[j], \cdots, G''_{k_2}[j]$ in Voting-Phase. Therefore, with a probability of at most $e^{-\Omega(k_1)} + e^{-\Omega(k_2)}$, $G'[j] \ne G[j]$. ▮

We will use multiple variable functions to characterize the computational time for three algorithms. In order to unify the complexity analysis of three algorithms, we introduce the following notation.

Definition 33. A function, $T(x, y) : N \times N \to N$, is nondecreasing if it is nondecreasing on both variables. If for arbitrary positive constants, c_1 and c_2, $T(c_1x, c_2y) \le cT(x, y)$ for some positive constant, c, then $T(x, y)$ is *slow*.

Lemma 34. *Assume that $t(x, y)$, $s(n, L)$ and $g(n, l)$ are non-decreasing slow functions. Assume that Collision-Detection(S_1, U_1, S_2, U_2) returns the result in $t(n, ||U_1|| + ||U_2||)$ time and the Point-Selection(S_1, S_2, L)) selects $s(n, L)$ positions in $g(n, L)$ time. Assume that with a probability of at most $\varphi(n)$, the function Initial-Boundaries() does not stop when $L \le |G|/4$, and $||U_{S'_{2i-1}}|| + ||U_{S''_j}||$ in the algorithm Recover-Motif is no more than $f(n, |G|)$.*

Then, with a probability of at most $k_1\varphi(n)$, the entire algorithm Recover-Motif does not stop in the time complexity $(O(k_1(\sum_{i=1}^{i_0}(t(n, s(n, \frac{n}{2^i n^{2/5}})) + g(n, \frac{n}{2^{i_0} n^{2/5}}))) + k_1h^2\log n + k_1k_2t(n, f(n, |G|)) + h^2\log n) + k_1k_2(\log n)(\log\log n)), O(k_2))$, where i_0 is the largest j, such that $\frac{n}{2^j n^{2/5}} \le \min(n^{2/5}, |G|)$ and

$h = \min(n^{2/5}, |G|)$.

Proof: The function Initial-Boundaries()is executed k_1 times. According to the condition that with a probability of at most $\varphi(n)$, the function Initial-Boundaries(.) does not stop when $L \le |G|/4$, we have the fact that with a probability of at most $k_1\varphi(n)$, one of those executions of Initial-Boundaries(.) does not stop when $L \le |G|/4$.

In the rest of the proof, we assume that all executions of Initial-Boundaries(.) stops when $L \le |G|/4$.

When $L = O(h)$, we detect rough left and right motif boundaries and run Improve-Boundaries(), which takes $O(h^2 \log n)$ time by Lemma 21. It takes $O(\sum_{i=1}^{i_0}(t(n, s(n, \frac{n}{2^i n^{2/5}})) + g(n, \frac{n}{2^i n^{2/5}}) + h^2\log n)$ time to run Initial-Boundaries(S'_{2i-1}, S'_{2i}) one time for one pair (S'_{2i-1}, S'_{2i}) in Z_1. It takes $O(k_1(\sum_{i=1}^{i_0}(t(n, s(n, \frac{n}{2^i n^{2/5}})) + g(n, \frac{n}{2^i n^{2/5}}) + k_1h^2\log n)$ time to run Initial-Boundaries(S'_{2i-1}, S'_{2i}) one time for all pairs (S'_{2i-1}, S'_{2i}) in Z_1.

It takes $k_2(t(n, f(n, |G|)) + h^2\log n)$ time to find the rough boundaries for all sequences in Z_2 with a fixed sequence, S, from Z_1 by executing for the loop "For each $S''_j \in Z_2$" in the algorithm Recover-Motif.

It takes $k_1k_2(t(n, f(n, |G|)) + h^2 \log n)$ time to find the rough boundaries for all sequences in Z_2 via all sequences, S'_{2i-1}, from Z_1 through the loop "For each $S''_j \in Z_2$" in the algorithm Recover-Motif.

Recall that parameters v and u_1 are constants, and u_2 is $O(\log\log n)$. Calling Match(G_l, G_r, S''_i) takes $O((v + u_2)\log n)$ time for each $S''_i \in Z_2$. The total times for calling Match(G_l, G_r, S''_i) is $O(k_1k_2(v + u_1)(v + u_2)\log n) = O(k_1k_2(\log n)(\log\log n))$.

The voting part takes $O(k_2)$ time for executing voting for recovering one character in the motif. ▮

6.4. Deterministic Algorithm for an $\Omega(1)$ Mutation Rate

In this section, we give a deterministic algorithm for the case with an $\Omega(1)$ mutation rate. The performance of the algorithm is stated in Theorem 6.

Lemma 35. *Let* algorithm-type=DETERMINISTIC-SUPERQUADRATIC. *Assume that* $d_0 \log n \leq L \leq |G|/2$ *and* $c_0 \log n \leq |G|$. *Let* I_1 *be a set of intervals of the positions of* S_1 *that satisfies* $[\text{LB}(S_1) - L, \text{LB}(S_1) + L] \cup [\text{RB}(S_1) - L, \text{RB}(S_1) + L] \subseteq \cup_{I' \in I_1} I'$. *Let* I_2 *be a set of intervals of the positions of* S_2 *that satisfies* $[\text{LB}(S_2) - L, \text{LB}(S_2) + L] \cup [\text{RB}(S_2) - L, \text{RB}(S_2) + L] \subseteq \cup_{I' \in I_2} I'$. *Let* U_1 = *Point-Selection*(S_1, L, I_1), U_2 = *Point-Selection*(S_2, L, I_2) *and* $(L_{S_1}, R_{S_1}, L_{S_2}, R_{S_2})$ = *Collision-Detection*$(S_1, U_1, S_2, , U_2)$. *Then:*

i. *with a probability of at most* $\frac{1}{2^x n^3}$, *the left rough boundary,* L_{S_1}, *has at most* $d_0 \log n$ *distance from* $\text{LB}(S_1)$ *and the left rough boundary* L_{S_2} *has at most* $d_0 \log n$ *distance from* $\text{LB}(S_2)$.

ii. *with a probability of at most* $\frac{1}{2^x n^3}$, *the right rough boundary,* R_{S_1}, *has at most* $d_0 \log n$ *distance from* $\text{RB}(S_1)$ *and the right boundary of* R_{S_2} *has at most* $d_0 \log n$ *distance from* $\text{RB}(S_2)$.

Proof: For two sequences, S_1 and S_2, let $\aleph(S_a)$ be the subsequence, $S_a[i_a, j_a]$, for $a = 1, 2$. By Corollary 18, with a probability of at most $P_l = 2c^{d_0 \log n} \leq \frac{2}{5 \cdot 2^x n^3}$ (see inequality (8) in Definition 14), there are more than $(\alpha + \epsilon)d_0 \log n$ mutations in $S_a[i_a, i_a + d_0 \log n - 1]$ for $a = 1, 2$.

With a probability of at most P_l, the left boundary position is missed during the matching. We have similar $P_r = P_l$ to miss the right boundary.

Assume that p_1 and p_2 are two positions of S_1 and S_2, respectively. If one of two positions is outside the motif region and has more than $d_0 \log n$ distance to the motif boundary, with a probability of at most $c^{-d_0 \log n} \leq \frac{1}{5 \cdot 2^x n^3}$ (see inequality (8) in Definition 14), for them to match requires $\text{diff}(Y_1, Y_2) \leq \beta$ by Lemma 19, where Y_a is a subsequence $S_a[p_a, p_a + d_0 \log n - 1]$ for $a = 1, 2$. ▮

Lemma 36. *For the case* algorithm-type=DETERMINISTIC-SUPERQUADRATIC, *we have:*

i. *Collision-Detection(*S_1, U_1, S_2, U_2*) takes* $t(n, ||U_1|| + ||U_2||) = O((||U_1|| + ||U_2||)^2 \log n)$ *time.*

ii. *Point-Selection(*$S_1, L, [1, |S_1|]$*) selects* $s(n, L) = O(n)$ *positions in* $g(n, L) = O(n)$ *time.*

iii. $||U_{S'_{2i-1}}|| + ||U_{S''_j}||$ *in the algorithm Recover-Motif is no more than* $f(n, |G|) = O(|G| + n)$.

iv. *With a probability of at most* $\frac{k}{2^x n^3}$, *the algorithm Recover-Motif does not run in computational complexity* $(O(k(n^2(\log n)^{O(1)} + h^2 \log n)), O(k))$.

Proof: Statement i. The parameter, $\omega_{\text{DETERMINISTIC-SUPERQUADRATIC}}$, is set to be β in Collision-Detection. It follows from the time complexity of the brute force method.

Statement ii. It follows from the implementation of Point-Selection().

Statement iii. It follows from the choice of Point-Selection(.) for the sublinear time algorithm at Recover-Motif(.).

Statement iv. It follows from Lemma 34 and Statements i, ii and iii. ▮

We give the proof for Theorem 6.

Proof: [Theorem 6] The computational time part of this theorem follows from Lemma 36.

By Lemma 35, we let $\varsigma_1(n) = \frac{1}{2^x n^3} \le \varsigma_0$ for the probability bound, $\varsigma_1(n)$, in the condition (i) of Lemma 32.

By Lemma 35, we can let $\varsigma_2(n) = \frac{1}{2^x n^3} \le \varsigma_0$ for the probability bound, $\varsigma_2(n)$, in the condition (ii) of Lemma 32.

By inequality (12), the condition (iii) of Lemma 32 is satisfied.

By inequality (11), we know that the condition (iv) of Lemma 32 can be satisfied.

The failure probability part of this theorem follows from Lemma 21 and Lemma 32 by using the fact that k_1, k_2 and k are of the same order (see equation (18)). ▮

6.5. Randomized Algorithms for Motif Detection

In this section, we present two randomized algorithms for motif detection. The first one is a sublinear time algorithm that can handle $\frac{1}{(\log n)^{2+\mu}}$ mutation, and the second one is a super-linear time algorithm that can handle $\Omega(1)$ mutation. They also share some common functions.

Lemma 37. *Let c_1 be a constant in $(0, 1)$. Assume m and n are two non-negative integers with $m \le n$. Then, for every integer, m_1, with $0 \le m_1 \le \frac{\delta_{c_1} m}{\ln n}$, $\binom{n}{m_1} c_1^m \le e^{(m \ln c_1)/2}$, where constant $\delta_{c_1} = \frac{-\ln c_1}{2}$ as defined in Definition 8.*

Proof: We have the inequalities:

$$\begin{aligned}
\binom{n}{m_1} c_1^m &\le n^{m_1} c_1^m && (23)\\
&= e^{m_1 \ln n} c_1^m && (24)\\
&\le e^{\frac{\delta_{c_1} m}{\ln n} \ln n} c_1^m && (25)\\
&= e^{\delta_{c_1} m} e^{m \ln c_1} && (26)\\
&= e^{(m \ln c_1)/2} && (27)
\end{aligned}$$

▮

Lemma 38. *Let $S = U \cup V$ be a set of n elements with $U \cap V = \emptyset$. Assume that $x_1, \cdots, x_m$ are m random elements in S. Then, with a probability of at most $\binom{||U||}{m_1}(\frac{||V||+m_1}{n})^m$, the list, $x_1, \cdots, x_m$, contains at most m_1 different elements from U (in other words, $||\{x_1, \cdots, x_m\} \cap U|| \le m_1$).*

Proof: For a subset, $S' \subseteq S$, with $||S'|| = m_0$ for some integer, $m_0 \geq 0$, the probability is at most $(\frac{m_0}{n})^m$ that all elements, $x_1, \cdots, x_m$, are in S'. For every subset, $X \subseteq S$, with $||X|| \leq m_1$, there exists another subset, $S' \subseteq S$, such that $||S'|| = m_1$ and $X \subseteq S'$. We have that $\Pr[||\{ x_1, \cdots, x_m\} \cap U|| \leq m_1] \leq \Pr[\{x_1, \cdots, x_m\} \cap U \subseteq U'$ for some $U' \subseteq U$ with $||U'|| = m_1]$. There are $\binom{||U||}{m_1}$ subsets of U with a size of m_1. We have the probability of at most $\binom{||U||}{m_1}(\frac{||V||+m_1}{n})^m$ that $x_1, \cdots, x_m$ contains at most m_1 different elements in U. ▮

Lemma 39. *Let β be a constant in $(0, 1)$ and $c_1 = 1 - \frac{\beta}{2}$. Let $m_1 \leq \frac{\delta_{c_1} m}{\ln \beta n}$ and $m \leq n^{1-\epsilon}$ for some fixed $\epsilon > 0$. Let S_1 and S_2 be two sets of n elements with $||S_1 \cap S_2|| \geq \beta n$ and C be a set of the size $||C|| \leq \gamma m_1$ for some constant $\gamma \in (0, 1)$. Then, for all large n, with a probability of at most $2e^{-\frac{(1-\gamma)m_1 m}{n}}$, we have $(A - C) \cap (B - C) = \emptyset$, where $A = \{x_1, \cdots, x_m\}$ and $B = \{y_1, \cdots, y_m\}$ are two sets, which may have multiplicities, of m random elements from S_1 and S_2, respectively.*

Proof: In the entire proof of this lemma, we always assume that n is sufficiently large. We are going to give an upper bound about the probability that B does not contain any element in $A - -C$. For each element, $y_i \in B$, with a probability of at most $1 - \frac{||A||-||C||}{n}$, y_i is not in $A - -C$. Therefore, the probability is at most $(1 - \frac{||A||-||C||}{n})^m$ that B does not contain any element in $A - -C$.

By Lemma 38, the probability is at most $\binom{\beta n}{m_1}(\frac{(1-\beta)n+m_1}{n})^m$ that $||A \cap (S_1 \cap S_2)|| \leq m_1$. We have the inequalities:

$$\begin{aligned}
& \Pr[(A-C) \cap (B-C) = \emptyset] && (28)\\
= \;& \Pr[(A-C) \cap (B-C) = \emptyset| \;\; ||A \cap (S_1 \cap S_2)|| \geq m_1] \cdot \Pr[||A \cap (S_1 \cap S_2)|| \geq m_1] + && (29)\\
& \Pr[(A-C) \cap (B-C) = \emptyset| \;\; ||A \cap (S_1 \cap S_2)|| < m_1] \cdot \Pr[|A \cap (S_1 \cap S_2)| < m_1] && (30)\\
\leq \;& \Pr[(A-C) \cap (B-C) = \emptyset| \;\; ||A \cap (S_1 \cap S_2)|| \geq m_1] + \Pr[||A \cap (S_1 \cap S_2)|| < m_1] && (31)\\
\leq \;& (1 - \frac{||A|| - ||C||}{n})^m + \binom{\beta n}{m_1}(\frac{(1-\beta)n + m_1}{n})^m && (32)\\
\leq \;& (1 - \frac{||(A \cap S_1 \cap S_2)|| - ||C||}{n})^m + \binom{\beta n}{m_1}(\frac{(1-\beta)n + m_1}{n})^m && (33)\\
\leq \;& (1 - \frac{(1-\gamma)m_1}{n})^m + \binom{\beta n}{m_1}(\frac{(1-\beta)n + m_1}{n})^m && (34)\\
\leq \;& e^{-\frac{(1-\gamma)m_1 m}{n}} + \binom{\beta n}{m_1}(\frac{(1-\beta)n + m_1}{n})^m && (35)\\
\leq \;& e^{-\frac{(1-\gamma)m_1 m}{n}} + \binom{\beta n}{m_1}(1 - \frac{\beta}{2})^m && (36)\\
\leq \;& e^{-\frac{(1-\gamma)m_1 m}{n}} + e^{(m \ln c_1)/2} && (37)\\
\leq \;& 2e^{-\frac{(1-\gamma)m_1 m}{n}} && (38)
\end{aligned}$$

The inequality, $(1 - \frac{(1-\gamma)m_1}{n})^m \leq e^{-\frac{(1-\gamma)m_1 m}{n}}$, which is used from Equation (34) to Equation (35), follows from the fact that $1 - x \leq e^{-x}$. The transition from (35) to (36) follows from the fact that $\frac{m_1}{n} \leq \frac{\beta}{2}$, since $m_1 = o(n)$, according to the conditions of the lemma.

It is easy to see that $\frac{2(1-\gamma)m_1 m}{-m \ln c_1} = \frac{2(1-\gamma)m_1}{-\ln c_1} \leq n$ for all large n. Thus, $\frac{(1-\gamma)m_1 m}{n} \geq (m \ln c_1)/2$ (note that $\ln c_1 < 0$ as $c_1 \in (0, 1)$). Thus, by Lemma 37, $\binom{\beta n}{m_1}(1 - \frac{\beta}{2})^m \leq e^{m \ln c_1/2} \leq e^{-\frac{(1-\gamma)m_1 m}{n}}$. This is why we have the transition from Equation (37) to Equation (38). Therefore, $\Pr[(A - C) \cap (B - C) = \emptyset] \leq 2e^{-\frac{(1-\gamma)m_1 m}{n}}$. ∎

6.5.1. . Randomized Algorithm for an $\Omega(1)$ Mutation Rate

In this section, we give an algorithm for the case with an $\Omega(1)$ mutation rate. The performance of the algorithm is stated in Theorem 4.

Lemma 40. *Let* algorithm-type=RANDOMIZED-SUBQUADRATIC. *Assume that* $d_0 \log n \leq L \leq |G|/2$ *and* $c_0 \log n \leq |G| < \frac{(\log n)^{3+\tau}}{100}$. *Let* I_1 *be a set of intervals of the positions of* S_1 *that satisfy* $[\mathrm{LB}(S_1) - L, \mathrm{LB}(S_1) + L] \cup [\mathrm{RB}(S_1) - L, \mathrm{RB}(S_1) + L] \subseteq \cup_{I' \in I_1} I'$. *Let* I_2 *be a set of intervals of the positions of* S_2 *that satisfy* $[\mathrm{LB}(S_2) - L, \mathrm{LB}(S_2) + L] \cup [\mathrm{RB}(S_2) - L, \mathrm{RB}(S_2) + L] \subseteq \cup_{I' \in I_2} I'$. *Let* U_1 = *Point-Selection*(S_1, L, I_1), U_2 = *Point-Selection*(S_2, L, I_2) *and* $(L_{S_1}, R_{S_1}, L_{S_2}, R_{S_2})$ = *Collision-Detection*$(S_1, U_1, S_2, , U_2)$. *Then:*

i. with a probability of at most $\frac{1}{2^x n^3}$, *the left rough boundary,* L_{S_1}, *has at most* $d_0 \log n$ *distance from* $\mathrm{LB}(S_1)$ *and the left rough boundary,* L_{S_2}, *has at most* $d_0 \log n$ *distance from* $\mathrm{LB}(S_2)$*;*

ii. with a probability of at most $\frac{1}{2^x n^3}$, *the right rough boundary,* R_{S_1}, *has at most* $d_0 \log n$ *distance from* $\mathrm{RB}(S_1)$ *and the right boundary of* R_{S_2} *has at most* $d_0 \log n$ *distance from* $\mathrm{RB}(S_2)$.

Proof: The proof is the same as Lemma 35 for the algorithm with algorithm-type=DETERMINISTIC-SUPERQUADRATIC. For two sequences, S_1 and S_2, let $\aleph(S_a)$ be the subsequence $S_a[i_a, j_a]$ for $a = 1, 2$. By Corollary 18, with a probability of at most $P_l = 2c^{d_0 \log n} \leq \frac{2}{5 \cdot 2^x n^3}$ (see inequality (8) in Definition 14), there are more than $(\alpha + \epsilon) d_0 \log n$ mutations in $S_a[i_a, i_a + d_0 \log n - 1]$ for $a = 1, 2$.

With a probability of at most P_l, the left boundary position is missed during the matching. We have similar $P_r = P_l$ to miss the right boundary.

Assume that p_1 and p_2 are two positions of S_1 and S_2, respectively. If one of the two positions is outside the motif region and has more than $d_0 \log n$ distance to the motif boundary, with a probability of at most $c^{-d_0 \log n} \leq \frac{1}{5 \cdot 2^x n^3}$ (see inequality (8) in Definition 14), for them to match requires $\mathrm{diff}(Y_1, Y_2) \leq \beta$ by Lemma 19, where Y_a is a subsequence $S_a[p_a, p_a + d_0 \log n - 1]$ for $a = 1, 2$. ∎

Lemma 41. *Let* algorithm-type=RANDOMIZED-SUBQUADRATIC. *Assume that* $d_0 \log n \leq L \leq |G|/2$ *and* $|G| \geq \frac{(\log n)^{3+\tau}}{100}$. *Let* I_1 *be a set of intervals of the positions of* S_1 *that satisfy* $[\mathrm{LB}(S_1) - L, \mathrm{LB}(S_1) + L] \cup [\mathrm{RB}(S_1) - L, \mathrm{RB}(S_1) + L] \subseteq \cup_{I' \in I_1} I'$. *Let* I_2 *be a set of intervals of the positions of* S_2 *that satisfy* $[\mathrm{LB}(S_2) - L, \mathrm{LB}(S_2) + L] \cup [\mathrm{RB}(S_2) - L, \mathrm{RB}(S_2) + L] \subseteq \cup_{I' \in I_2} I'$. *Let* U_1 = *Point-Selection*(S_1, L, I_1), U_2 = *Point-Selection*$(S_2, L, I_2])$ *and* $(L_{S_1}, R_{S_1}, L_{S_2}, R_{S_2})$ = *Collision-Detection*(S_1, U_1, S_2, U_2). *Then:*

i. with a probability of at most $\frac{1}{2^x n^3}$, *the left rough boundary,* L_{S_1}, *has at most a* $2L$ *distance from* $\mathrm{LB}(S_1)$ *and the left rough boundary* L_{S_2} *has at most a* $2L$ *distance from* $\mathrm{LB}(S_2)$*;*

ii. with a probability of at most $\frac{1}{2^x n^3}$, *the right rough boundary,* R_{S_1}, *has at most a* $2L$ *distance from* $\mathrm{RB}(S_1)$ *and the right boundary of* R_{S_2} *has at most a* $2L$ *distance from* $\mathrm{RB}(S_2)$.

Proof: We prove the following two statements which imply the lemma.

i. With a probability of at most $\frac{1}{2^x n^3}$, there are no intervals, A_i from S_1 and B_j from S_2, such that: (1) $||A_i(S_1, \aleph(S_1)) \cap B_j(S_2, \aleph(S_2))||$ is at least $\frac{L}{2}$; (2) the left boundary of S_1 has at most a $2L$ distance from A_i; (3) the left boundary of S_2 has at most a $2L$ distance from B_j; and (4) there is a collision between the sampled positions in A_i and B_j.

ii. With a probability pf at most $\frac{1}{2^x n^3}$, there are no intervals, A_i from S_1 and B_j from S_2, such that: (1) $||A_i(S_1, \aleph(S_1)) \cap B_j(S_2, \aleph(S_2))||$ is at least $\frac{L}{2}$; (2) the right boundary of S_1 has at most a $2L$ distance from A_i; (3) the right boundary of S_2 has at most a $2L$ distance from B_j; and (4) there is a collision between the sampled positions in A_i and B_j.

We only prove statement i. The proof for statement ii is similar.

Select A_i from S_1 and B_j from S_2 to be the first pair of intervals with $||A_i(S_1, \aleph(S_1)) \cap B_j(S_2, \aleph(S_2))|| \geq \frac{L}{2}$. It is easy to see that such a pair exists, and both have a distance from the left boundary with a distance of at most $2L$. This is because when a leftmost interval of a length pf L is fully inside the motif region of the first sequence, we can always find the second interval from the second sequence with an intersection of a length of at least $\frac{L}{2}$.

Replace m by $M(L)$, m_1 by $M_1(L)$ (see Definition 10) and n by L to apply Lemma 39. We do not consider any damaged position in this algorithm; therefore, let C be empty.

With a probability of at most $o(\frac{1}{2^x n^3})$, there is a point in $(A_i(S_1, \aleph(S_1)) - C) \cap (B_j(S_2, \aleph(S_2)) - C)$. The subsequences of a length of $d_0 \log n$ starting at the point from S_1 and S_2 fail to have the difference bounded by β with a probability of at most $o(\frac{1}{2^x n^3})$ by Lemma 20. With a probability of at most $o(\frac{1}{2^x n^3})$, we do not have that the rough boundaries are detected with a distance of at most $2L$ to exact motif boundaries. ∎

Lemma 42. *For the case* algorithm-type=RANDOMIZED-SUBQUADRATIC, *we have:*

i. Collision-Detection(S_1, U_1, S_2, U_2) takes $t(n, ||U_1|| + ||U_2||) = O((||U_1|| + ||U_2||)^2 \log n)$ time.

ii. Point-Selection($S_1, L, [1, |S_1|]$) selects $s(n, L) = O((\frac{n}{L})M(L))$ positions in $g(n, L) = O(s(n, L))$ time if $L \geq \frac{(\log n)^{3+\tau}}{100}$.

iii. Point-Selection($S_1, L, [1, |S_1|]$) selects $s(n, L) = O(n)$ positions in $g(n, L) = O(n)$ time if $L < \frac{(\log n)^{3+\tau}}{100}$.

iv. $||U_{S'_{2i-1}}|| + ||U_{S''_j}||$ in the algorithm Recover-Motif is no more than $f(n, |G|) = O(M(|G|) + \frac{n}{|G|}M(|G|))$.

v. With a probability of at most $\frac{k}{2^x n^3}$, the algorithm Recover-Motif does not stop in $(O(k(\frac{n^2}{|G|}(\log n)^{O(1)} + h^2 \log n)), O(k))$ time.

Proof: Statement i. The parameter, $\omega_{\text{RANDOMIZED-SUBLINEAR}}$, is set to be β in Collision-Detection. It follows from the time complexity of the brute force method.

Statements ii and iii. They follow from the implementation of Point-Selection().

Statement iv. It follows from the choice of Point-Selection(.) for the sublinear time algorithm at Recover-Motif(.).

Statement iv. It follows from Lemma 40, Lemma 34 and Statements i, ii and iii. ∎

We give the proof for Theorem 6.

Proof: [Theorem 4] The computational time part of this theorem follows from Lemma 42.

By Lemma 40, we can let $\varsigma_1(n) = \frac{1}{2^x n^3} \le \varsigma_0$ for the probability bound, $\varsigma_1(n)$, in the condition (i) of Lemma 32.

By Lemma 41, we can let $\varsigma_2(n) = \frac{1}{2^x n^3} \le \varsigma_0$ for the probability bound, $\varsigma_2(n)$, in the condition (ii) of Lemma 32.

By inequality (12), the condition (iii) of Lemma 32 is satisfied.

By inequality (11), we know that the condition (iv) of Lemma 32 can be satisfied.

The failure probability part of this theorem follows from Lemma 21 and Lemma 32 by using the fact that k_1, k_2 and k are of the same order (see Equation (18)). ∎

6.5.2. . Sublinear Time Algorithm for a $\frac{1}{(\log n)^{2+\mu}}$ Mutation Rate

In this section, we give an algorithm for the case with at most $\alpha = \frac{1}{(\log n)^{2+\mu}}$ mutation rate. The performance of the algorithm is stated in Theorem 2.

Definition 43. A position, p, in the motif region, $\aleph(S)$, of an input sequence, S, is *damaged* if there exists at least one mutation in $S[p, p + d_0 \log n - 1]$, where d_0 is defined in item (xvii) in Definition 8.

Lemma 44. *Assume that* $\alpha L = (\log n)^{1+\Omega(1)}$. *With a probability of at most* $e^{-(\log n)^{1+\Omega(1)}}$, *there are more than* $\frac{M_1(L)}{(\log n)^{\Omega(1)}}$ *positions that are from the* $M(L)$ *(see Definition 10 for* $M(.)$ *and* $M_1(.)$*) sampled positions that are damaged in an interval of a length pf* L.

Proof: By Theorem 16, with a probability of at most $P_1 = 2^{-\alpha L}$ (let $\delta = 2$), there are more than $3\alpha L$ mutations in an interval of a length of L. Therefore, with a probability of at most $2^{-\alpha L} = e^{-(\log n)^{1+\Omega(1)}}$, there are more than $3\alpha L \log n$ positions that are damaged. Therefore, each random position in an interval of a length of L has at most a probability of $\frac{3\alpha L \log n}{L} = 3\alpha \log n$ to be damaged.

Since $\alpha = (\frac{1}{(\log n)^{2+\Omega(1)}})$ and $M(L)$ positions are sampled, by Theorem 16, with a probability of at most $P_2 = 2^{-(3\alpha \log n)M(L)} = e^{-(\log n)^{1+\Omega(1)}}$ (let $\delta = 2$), the number of damaged positions sampled in an interval of a length of L is more than $((1+\delta)3\alpha \log n)M = (9\alpha \log n)M(L) = \frac{M_1(L)}{(\log n)^{\Omega(1)}}$. Thus, with a total probability of at most $P_1 + P_2 = e^{-(\log n)^{1+\Omega(1)}}$, there are more than $\frac{M_1(L)}{(\log n)^{\Omega(1)}}$ damaged positions that are from the $M(L)$ sampled positions in an interval of a length of L. ∎

Definition 45. Let A be a set of positions in an input sequence, S, with $\aleph(S) = S[i,j]$. Let $A(S, \aleph(S)) = \{x - i + 1 | x \in A \cap [i,j]\}$. In other words, $A(S, \aleph(S))$ contains all the positions of A in $\aleph(S)$.

Lemma 46. *Let* algorithm-type=RANDOMIZED-SUBLINEAR. *Assume that* $|G| < \frac{(\log n)^{3+\tau}}{100}$ *and* L *is an integer with* $d_0 \log n \leq L \leq |G|/2$. *Let* I_1 *be a set of intervals of the positions of* S_1 *that satisfy* $[\mathrm{LB}(S_1) - L, \mathrm{LB}(S_1) + L] \cup [\mathrm{RB}(S_1) - L, \mathrm{RB}(S_1) + L] \subseteq \cup_{I' \in I_1} I'$. *Let* I_2 *be a set of intervals of the positions of* S_2 *that satisfy* $[\mathrm{LB}(S_2) - L, \mathrm{LB}(S_2) + L] \cup [\mathrm{RB}(S_2) - L, \mathrm{RB}(S_2) + L] \subseteq \cup_{I' \in I_2} I'$. *Let* U_1 = *Point-Selection*(S_1, L, I_1), U_2 = *Point-Selection*$(S_2, L, I_2])$ *and* $(L_{S_1}, R_{S_1}, L_{S_2}, R_{S_2})$ = *Collision-Detection*(S_1, U_1, S_2, U_2). *Then:*

i. *with a probability of at most* $\frac{1}{2^x n^3}$, *the left rough boundary,* L_{S_1}, *has at most a* $|G|/4$ *distance from* $\mathrm{LB}(S_1)$ *and the left rough boundary,* L_{S_2}, *has at most a* $|G|/4$ *distance from* $\mathrm{LB}(S_2)$.

ii. *with a probability of at most* $\frac{1}{2^x n^3}$, *the right rough boundary,* R_{S_1}, *has at most a* $|G|/4$ *distance from* $\mathrm{RB}(S_1)$ *and the right boundary of* R_{S_2} *has at most a* $|G|/4$ *distance from* $\mathrm{RB}(S_2)$.

Proof: For two sequences, S_1 and S_2, it is easy to see that there is a common position in both motif regions of the two sequences, such that there is no mutation in the next $d_0 \log n$ characters with a high probability. This is because that mutation probability is small.

By Theorem 16, with a probability of at most $P_l = 2^{-\alpha|G|/4}$ (let $\delta = 2$), there are more than $3\alpha\frac{|G|}{4}$ mutated characters in the interval $\aleph(S_i)[1, \frac{|G|}{4}]$ for $i = 1, 2$. Since the mutation probability is $\alpha = (\frac{1}{(\log n)^{2+\Omega(1)}})$, with a probability of at most $2^{-\alpha|G|/4} = e^{-(\log n)^{1+\Omega(1)}}$, there are more than $3\alpha \cdot \frac{|G|}{4} d_0 \log n$ positions to be damaged in $\aleph(S_i)[1, \frac{|G|}{4}]$.

Similarly, we have a probability of at most $P_r = e^{-(\log n)^{1+\Omega(1)}}$ that there are more than $((3\alpha d_0 \log n) \cdot \frac{|G|}{4}) = \frac{|G|}{(\log n)^{\Omega(1)}}$ damaged positions in $\aleph(S_i)[\frac{3|G|}{4} - 1, |G|]$.

Now, we assume that left side has more than $((3\alpha d_0 \log n) \cdot \frac{|G|}{4}) = \frac{|G|}{(\log n)^{\Omega(1)}}$ damaged positions and the right side, more than $((3\alpha d_0 \log n) \cdot \frac{|G|}{4}) = \frac{|G|}{(\log n)^{\Omega(1)}}$ damaged positions in $\aleph(S_i)[\frac{3|G|}{4} - 1, |G|]$. Since each position in each interval of a length of L is selected in Point-Selection(S_1, S_2, L), it is easy to verify the conclusions of this lemma. ▮

Lemma 47. *Let* algorithm-type=RANDOMIZED-SUBLINEAR. *Assume that* $|G| \geq \frac{(\log n)^{3+\tau}}{100}$ *and* $d_0 \log n \leq L \leq |G|/2$. *Let* I_1 *be a set of intervals of the positions of* S_1 *that satisfy* $[\mathrm{LB}(S_1) - L, \mathrm{LB}(S_1) + L] \cup [\mathrm{RB}(S_1) - L, \mathrm{RB}(S_1) + L] \subseteq \cup_{I' \in I_1} I'$. *Let* I_2 *be a set of intervals of the positions of* S_2 *that satisfy* $[\mathrm{LB}(S_2) - L, \mathrm{LB}(S_2) + L] \cup [\mathrm{RB}(S_2) - L, \mathrm{RB}(S_2) + L] \subseteq \cup_{I' \in I_2} I'$. *Let* U_1 =*Point-Selection*(S_1, L, I_1), U_2 =*Point-Selection*$(S_2, L, I_2])$ *and* $(L_{S_1}, R_{S_1}, L_{S_2}, R_{S_2})$ =*Collision-Detection*(S_1, U_1, S_2, U_2). *Then:*

i. *with a probability of at most* $\frac{1}{2^x n^3}$, *the left rough boundary,* L_{S_1}, *has at most a* $2L$ *distance from* $\mathrm{LB}(S_1)$ *and the left rough boundary,* L_{S_2}, *has at most a* $2L$ *distance from* $\mathrm{LB}(S_2)$.

ii. *With a probability of at most* $\frac{1}{2^x n^3}$, *the right rough boundary,* R_{S_1}, *has at most a* $2L$ *distance from* $\mathrm{RB}(S_1)$ *and the right boundary of* R_{S_2} *has at most a* $2L$ *distance from* $\mathrm{RB}(S_2)$.

Proof: We prove the following two statements, which imply the lemma.

i. With a probability of at most $\frac{1}{2^x n^3}$, there are no intervals, A_i from S_1 and B_j from S_2, such that: (1) $||A_i(S_1, \aleph(S_1)) \cap B_j(S_2, \aleph(S_2))||$ is at least $\frac{L}{2}$; (2) the left boundary of S_1 has at most a $2L$ distance from A_i; (3) the left boundary of S_2 has at most a $2L$ distance from B_j; and (4) there is a collision between the sampled positions in A_i and B_j.

ii. with a probability of at most $\frac{1}{2^x n^3}$, there are no intervals, A_i from S_1 and B_j from S_2, such that: (1) $||A_i(S_1, \aleph(S_1)) \cap B_j(S_2, \aleph(S_2))||$ is at least $\frac{L}{2}$; (2) the right boundary of S_1 has at most a $2L$ distance from A_i; (3) the right boundary of S_2 has at most a $2L$ distance from B_j; and (4) there is a collision between the sampled positions in A_i and B_j.

We only prove statement i. The proof for statement ii is similar to that for statement i. Assume that L satisfies the condition of this lemma. Select A_i from S_1 and B_j from S_2 to be the first pair of intervals of a size of L with $||A_i(S_1, \aleph(S_1)) \cap B_j(S_2, \aleph(S_2))|| \geq \frac{L}{2}$. This is because when a leftmost interval of a length of L is fully inside the motif region of the first sequence, we can always find the second interval from the second sequence with an intersection of a length of at least $\frac{L}{2}$.

Replace m by $M(L)$, m_1 by $M_1(L)$ (see Definition 10) and n by L to apply Lemma 39. We also let C be the set of damaged positions in $\aleph(S_1)$ and $\aleph(S_2)$ caused by the mutated positions. With a probability of at most $o(\frac{1}{2^x n^3})$, C has a size of $o(M_1(L))$ by Lemma 44. With a probability of at most $o(\frac{1}{2^x n^3})$, there is point in $(A_i(S_1, \aleph(S_1)) - C) \cap (B_j(S_2, \aleph(S_2)) - C)$. The existence of such a point makes L_{S_1} and L_{S_2} have a distance of at most $2L$ to $\mathrm{LB}(S_1)$ and $\mathrm{LB}(S_2)$, respectively. ∎

Lemma 48. *For the case* algorithm-type=RANDOMIZED-SUBLINEAR, *we have:*

i. Collision-Detection(S_1, U_1, S_2, U_2*) takes* $t(n, ||U_1|| + ||U_2||) = O((||U_1|| + ||U_2||) \log n)$ *time.*

ii. Point-Selection($S_1, L, [1, |S_1|]$*) selects* $s(n, L) = O((\frac{n}{L})M(L))$ *positions in* $g(n, L) = O(s(n, L))$ *time if* $L \geq \frac{(\log n)^{3+\tau}}{100}$.

iii. Point-Selection($S_1, L, [1, |S_1|]$*) selects* $s(n, L) = O(n)$ *positions in* $g(n, L) = O(n)$ *time if* $L < \frac{(\log n)^{3+\tau}}{100}$.

iv. $||U_{S'_{2i-1}}|| + ||U_{S''_j}||$ *in the algorithm Recover-Motif is no more than* $f(n, |G|) = O(M(|G|) + \frac{n}{|G|}M(|G|))$.

v. With a probability of at most $\frac{k}{2^x n^3}$, *the algorithm Recover-Motif does not stop in* $(O(k(\frac{n}{\sqrt{h}}(\log n)^{\frac{5}{2}} + h^2 \log n)), O(k))$ *time.*

Proof: Statement i. The parameter, $\omega_{\mathrm{RANDOMIZED-SUBLINEAR}}$, is set to be zero in Collision-Detection. It follows from the time complexity of bucket sorting, which is described in standard algorithm textbooks.

Statements ii and iii. They follow from the implementation of Point-Selection().

Statement iv. It follows from the choice of Point-Selection(.) for the sublinear time algorithm at Recover-Motif(.).

Statement v. It follows from Lemma 46, Lemma 47 and Lemma 34 and Statements i, ii iii and iv. ∎

We give the proof for Theorem 2.

Proof: [Theorem 2] The computational time part of this theorem follows from Lemma 48.

By Lemma 46, we can let $\varsigma_1(n) = \frac{1}{2^x n^3} \leq \varsigma_0$ for the probability bound, $\varsigma_1(n)$, in the condition (i) of Lemma 32.

By Lemma 47, we can let $\varsigma_2(n) = \frac{1}{2^x n^3} \leq \varsigma_0$ for the probability bound, $\varsigma_1(n)$, in the condition (ii) of Lemma 32.

By inequality (12), the condition (iii) of Lemma 32 is satisfied.

By inequality (11), we know that the condition (iv) of Lemma 32 can be satisfied.

The failure probability part of this theorem follows from Lemma 21 and Lemma 32 by using the fact that k_1, k_2 and k are of the same order (see Equation (18)). ∎

6.6. Experiments on Simulated Datasets

Aiming at solving the motif discovery problem, we implemented our algorithm in Java. Our tests were all done on a laptop with an Intel Dual Core 1.5 G CPU and 3.0 G Memory. In the first experiment, we tested our algorithm on several simulated datasets, which are all generated from our probability model with a small mutation rate. Each input set contains 20 or 15 sequences, which are of a length of 600 or 500 base pairs. Additionally, each bp of all the simulated gene sequences was generated independently with the same occurrence probability. A motif with a fixed length was randomly planted into each input sequence. The minimum Hamming distances between the results and consensus are recorded.

6.7. Experiments

There are many other tools for detecting and analyzing the motifs, like the EMmethod [23], MEME [24], Gibbs [25], Compo [26], MochiView [27], PhyME [28], HeliCis [29] and WebMotif [30], among others. Each of them has their advantages and disadvantages.

Table 1 shows the results on simulated datasets. From the table, we could find that the results of our algorithm for finding the motif on simulated datasets are satisfied. Our algorithm could find all the motifs from each sequence and get the consensus with an accuracy rate of 100%. If the dataset has a high mutation rate, we could increase the number of repetitions, so that the result on the datasets will be more accurate. We also recorded the total time cost for each test, which mainly depends on the number of test repetitions. In the experimental tables, the parameter, N, represents the number of sequences, parameter M represents the length of motifs, parameter M is the maximum length of sequences and R is the number of iterations. GCR1 is a famous DNA binding protein, whose ability to bind DNA is dependent on the CTTCC sequence motif [31]. Several other popular data sets are also used in the experiments of our motif detection and its comparisons to the other methods.

Table 1. Results on simulated data.

	N	**M**	**L**	**R**	**Accuracy Rate**	**Time cost(s)**
Set 1	20	600	15	60	100	98
Set 2	15	600	15	10	100	18
Set 3	20	600	12	15	100	22
Set 4	20	500	15	40	100	79

In the second experiment, we tested our algorithm on a real sequence set, which was obtained from SCPD. SCPD contains a large number of gene data and transcription factors of yeast. Sequences in the same set are all regulated by a common motif. We chose 1,000 bp as the length of the input sequences. In order to show the advantages of our algorithm, we also compared the result of our algorithm with the results of several other existing motif finding methods on the same dataset, such as Gibbs, MEME, Info-Gibbs and Consensus. Table 2 shows the details of the data set we used in the experiment.

Table 2. Number of Sequences and Motif Length.

	Bas1	**GCN4**	**GCR1**	**Rap1Ebf1**	**HSE-HTSF**
N	6	9	6	15	5
L	10	10	10	15	10

Table 3 shows the results of the five algorithms on biological data sets. Table 4 and Figure 2 give the average mismatch numbers of each algorithm. We choose four well-known motif-detecting softwares to make the comparisons. From Table 3 and 4, we see that the average mismatch numbers of our algorithm on data sets GCN4 and GCR1 are significantly lower than other four well-known methods. On the data sets Bas1,Rap1Ebf1 and HSE-HTSF, our algorithms also shows satisfied performance compared to other methods.

In addition, our algorithm also shows its high speed in computations compared to other four motif finding methods. Because the starting pattern of algorithms are represented by a string, so our algorithm can avoid some extra time consuming computations unlike Gibbs sampling and EM methods, such as computations of likelihoods. According to this feature, we use the consensus string of the voting operation obtained from the last iteration as a new starting pattern in program, and continue doing voting operations repeatedly until there is no further improvement. Experimental results show that if we set the number of iterations to be a large integer, the programs could give more accurate results within a reasonable time.

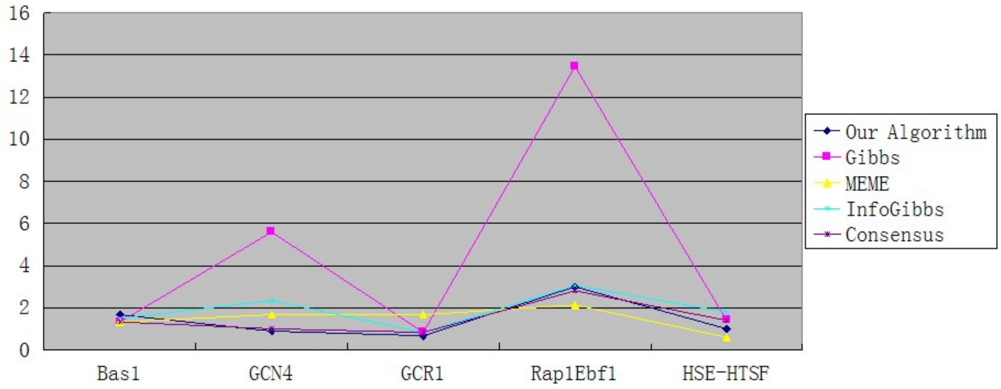
Table 3. Total number of mismatch positions.

	Bas1	**GCN4**	**GCR1**	**Rap1Ebf1**	**HSE-HTSF**
Our Algorithm	10	8	4	45	5
Gibbs	8	51	5	202	7
MEME	8	15	10	32	3
InfoGibbs	9	21	5	46	9
Consensus	8	9	5	42	7

Table 4. Average mismatch numbers per sequence.

	Bas1	**GCN4**	**GCR1**	**Rap1Ebf1**	**HSE-HTSF**
Our Algorithm	1.67	0.89	0.67	3	1
Gibbs	1.33	5.6	0.83	13.46	1.4
MEME	1.33	1.67	1.67	2.13	0.6
InfoGibbs	1.5	2.33	0.83	3.06	1.8
Consensus	1.33	1	0.83	2.8	1.4

Figure 2. Average mismatch numbers per sequence.



6.8. Conclusions

We develop algorithms under the probabilistic model. One of them finds the implanted motif with a high probability if the alphabet size is at least four, the motif length is in $[(\log n)^{7+\mu}, \frac{n}{(\log n)^{1+\mu}}]$ and each character in the motif region has a probability of at most $\frac{1}{(\log n)^{2+\mu}}$ of mutation. The motif region can be detected, and each motif character can be recovered in sublinear time. A sub-quadratic randomized algorithm is developed to recover the motif with an $\Omega(1)$ mutation rate. A quadratic deterministic algorithm is developed to recover the motif with an $\Omega(1)$ mutation rate. It is an interesting problem if there is an algorithm to handle the case for an alphabet of a size of three. A more interesting problem is to extend the algorithm to handle larger mutation probability.

6.9. Future Works

Compared with other motif finding methods, our algorithm shows its great advantages. However, there are still some improvements that could be done on this algorithm. For example, though a sequence set has the consensus, the motif in each sequence may have high mutation rates; in addition, the length of each motif could also be different. Therefore, these two factors increase the difficulty for finding motifs, and currently, there is still no effective algorithm that could solve these problems. In the future, we plan to improve the efficiency of our algorithm by combining other motif finding methods, such as MEME; a combination may be made to make our algorithm have better performance in finding motifs from highly mutated sequences.

Acknowledgments

We thank Ming-Yang Kao for introducing us to this topic. We also thank Lusheng Wang and Xiaowen Liu for some discussions. We would like to thank Eugenio De Hayos for his helpful comments. We would like to thank the reviewers, whose comments greatly improved the presentation of this paper.

This research is supported in part by the NSF Early Career Award 0845376, and NSF HRD-1137764.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Frances, M.; Litman, A. On covering problems of codes. *Theor. Comput. Sci.* **1997**, *30*, 113–119.
2. Gąsieniec, L.; Jansson, J.; Lingas, A. Efficient Approximation Algorithms for the Hamming Center Problem. In Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, USA, 17–19 January 1999; pp. S905–S906.
3. Stormo, G.; Hartzell , G., III. Identifying protein-binding sites from unaligned DNA fragments. *Proc. Natl. Acad. Sci. USA* **1991**, *88*, 5699–5703.
4. Lawrence, C.; Reilly, A. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins* **1990**, *7*, 41–51.
5. Hertz, G.; Stormo, G. Identification of Consensus Patterns in Unaligned DNA and Protein Sequences: A Large-Deviation Statistical Basis for Penalizing Gaps. In Proceedings of the 3rd International Conference on Bioinformatics and Genome Research, Tallahassee, USA, 1–4 June 1994; pp. 201–216.
6. Stormo, G. Consensus patterns in DNA. In *Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences*; Doolitle, R.F., Ed.; *Methods Enzymol.* **1990**, *183*, 211–221.
7. Lanctot, J.K.; Li, M.; Ma, B.; Wang, L.; Zhang, L. Distinguishing string selection problems. *Inf. Comput.* **2003**, *185*, 41–55.
8. Lucas, K.; Busch, M.; Mossinger, S.; Thompson, J. An improved microcomputer program for finding gene- or gene family-specific oligonucleotides suitable as primers for polymerase chain reactions or as probes. *Comput. Appl. Biosci.* **1991**, *7*, 525–529.
9. Dopazo, J.; Rodríguez, A.; Sáiz, J.C.; Sobrino, F. Design of primers for PCR amplification of highly variable genomes. *Comput. Appl. Biosci.* **1993**, *9*, 123–125.
10. Proutski, V.; Holme, E.C. Primer master: A new program for the design and analysis of PCR primers. *Comput. Appl. Biosci.* **1996**, *12*, 253–255.
11. Li, M.; Ma, B.; Wang, L. On The Closest String and Substring Problems. *J. ACM* **2002**, *49*, 157–171.
12. Li, M.; Ma, B.; Wang, L. Finding Similar Regions in Many Strings. In Proceedings of the 31st Annual ACM Symposium on Theory of Computing, Atlanta, GA, USA, 1–4 May 1999; pp. 473–482.

13. Pevzner, P.; Sze, S. Combinatorial Approaches to Finding Subtle Signals in DNA Sequences. In Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology, Toronto, ON, Canada, 19–23 July 2000; pp. 269–278.
14. Keich, U.; Pevzner, P. Finding motifs in the twilight zone. *Bioinformatics* **2002**, *18*, 1374–1381.
15. Keich, U.; Pevzner, P. Subtle motifs: Defining the limits of motif finding algorithms. *Bioinformatics* **2002**, *18*, 1382–1390.
16. Wang, L.; Dong, L. Randomized algorithms for motif detection. *J. Bioinform. Comput. Biol.* **2005**, *3*, 1039–1052.
17. Chin, F.; Leung, H. Voting Algorithms for Discovering Long Motifs. In Proceedings of the 3rd Asia-Pacific Bioinformatics Conference, Singapore, 17–21 January 2005; pp. 261–272.
18. Gusfield, D. *Algorithms on Strings, Trees, and Sequences*; Cambridge University Press: Cambridge, UK, 1997.
19. Fu, B.; Kao, M.Y.; Wang, L. Probabilistic analysis of a motif discovery algorithm for multiple sequences. *SIAM J. Discret. Math.* **2009**, *23*, 1715–173.
20. Fu, B.; Kao, M.Y.; Wang, L. Discovering almost any hidden motif from multiple sequences. ACM Transactions on Algorithms **2011**, *7(2)*, 26.
21. Liu, X.; Ma, B.; Wang, L. Voting Algorithms for the Motif Problem. In Proceedings of Computational Systems Bioinformatics Conference, (CSB'08), Stanford, CA, USA, 26–29 August 2008; pp. 37–47.
22. Motwani, R.; Raghavan, P. *Randomized Algorithms*; Cambridge University Press: Cambridge, UK, 2000.
23. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum likelihood from complete data vis the EM algorithm. *J. R. Stat. Soc.* **1977**, *39*, 1–38.
24. D'haesler, P. How does DNA sequence motif discovery work? *Nat. Biotechnol.* **2006**, *24*, 959–961.
25. Lawrence, C.; Altschul, S.; Boguski, M.; Liu, J.; Neuwald, A.; Wootton, J. Detecting subtle sequence signals: A gibbs sampling strategy for multiple alignment. *Science* **1993**, *262*, 262–5131.
26. Sandve, G.K.K.; Abul, O.; Drabløs, F. Compo: Composite motif discovery using discrete models. *BMC Bioinform.* **2008**, *9*, doi:10.1186/1471-2105-9-527.
27. Homann, O.; Johnson, A. MochiView: Versatile software for genome browsing and DNA motif analysis. *BMC Biol.* **2010**, *8*, doi:10.1186/1741-7007-8-49.
28. Sinha, S.; Blanchette, M.; Tompa, M. PhyME: A probabilistic algorithm for finding motifs in sets of orthologous sequences. *BMC Bioinform.* **2004**, *5*, doi:10.1186/1471-2105-5-170.
29. Larsson, E.; Lindahl, P.; Mostad, P. HeliCis: A DNA motif discovery tool for colocalized motif pairs with periodic spacing. *BMC Bioinform.* **2007**, *8*, doi:10.1186/1471-2105-8-418.
30. Romer, K.; Kayombya, G.R.; Fraenkel, E. WebMOTIFS: Automated discovery, filtering and scoring of DNA sequence motifs using multiple programs and Bayesian approaches. *Nucleic Acids Res.* **2007**, *35*, W217–W220.

31. Baker, H. GCR1 of Saccharomyces cerevisiae encodes a DNA binding protein whose binding is abolished by mutations in the CTTCC sequence motif. *Proc. Natl. Acad. Sci. USA* **1991**, *88*, 9443–9447.