Comparative Analysis of Motif Finding Algorithms

Hafijul Hoque Chowdhury
1905013

Syeda Rifah Tasfia
1905019

Sardar Md. Saffat Zabin
1905060

Anindya Hoque
1905070

Asad Bin Shahid Mahin
1905087

Abstract

Motif finding, a fundamental problem in bioinformatics, involves identifying recurring patterns in biological sequences that often represent functional elements such as transcription factor binding sites. In this report, we present a comparative analysis of two prominent motif finding algorithms: Randomized Motif Search (RMS) and Gibbs Sampler. Additionally, we evaluate the effectiveness of two widely used motif discovery tools, MEME and STREME, which implement these algorithms.

RMS employs a randomized strategy, iteratively searching for motifs within a set of sequences, while Gibbs Sampler adopts a probabilistic sampling approach to refine motif predictions based on the posterior distribution of motif occurrences. MEME leverages probabilistic modeling to identify motifs enriched in the input sequences, while STREME focuses on discovering motifs enriched in specific subsets of sequences.

We experiment with different techniques and assess the performance of these methods over a number of different datasets based on computational efficiency and accuracy. Our comparative analysis provides insights into the strengths and limitations of each algorithm and tool in motif finding tasks.

1 Introduction

The motif finding problem involves identifying recurring patterns in biological sequences that often represent functional elements such as transcription factor binding sites. It is a fundamental problem in bioinformatics, with applications in gene regulation, protein-DNA interactions, and other biological processes. The discovery of motifs is crucial for understanding the regulatory mechanisms of genes and proteins, and has implications in drug discovery and disease treatment. It is an NP-hard problem, which is made difficult by a number of biological factors.

Motif finding involves searching for many small sequences among long DNA sequences. Moreover, motifs are degenerate, as there are variations in their nucleotide sequences caused during cell division and mutation. Data for DNA sequences is often noisy and contains errors, making it difficult to identify them. Additionally, motifs can be located in different positions in different sequences, and the length of the motif is often unknown. Each sequence may contain multiple motifs; moreover they may be overlapped or nested. Thus, heuristic algorithms are used to solve the motif finding problem.

In this report we present a comparative analysis of two prominent motif finding algorithms: Randomized Motif Search (RMS) and Gibbs Sampler. Additionally, we evaluate the effectiveness of two widely used motif discovery tools, MEME and STREME, which implement these algorithms.

2 Data

We evaluate the performance of the motif finding algorithms on a small set of real datasets, `hm03`, `yst04r` and `yst08r`. The first dataset, `hm03`, is derived from human genome, consisting of 10 sequences of 1500 bases. The other two datasets originate from yeast genome: the former, `yst04r`, containing 7 sequences of 1000 bases and the latter, `yst08r`, containing 11 sequences of 1000 bases. Thus, these datasets are composed of both mammalian and fungal samples, while containing different numbers of sequences. This allows us to assess the performance of the algorithms on a small yet diverse dataset.

3 Algorithms

We utilized two prominent motif finding algorithms, Randomized Motif Search (RMS) and Gibbs Sampler in our study. For both algorithms, we defined the score function as the sum of Hamming distances between the motifs and the consensus sequence.

3.1 Randomized Motif Search (RMS)

Randomized Motif Search (RMS) is a randomized algorithm that iteratively searches for motifs within a set of sequences. It is particularly useful for discovering short, recurring sequences that may represent regulatory motifs.

The steps of the RMS algorithm are as follows:

1. **Initialization** Randomly select k-mers from each sequence in the dataset. This forms the initial set of motifs.
2. **Motif Profile Construction** Construct a profile matrix from the selected motifs, consisting of the frequency of each nucleotide at each position in the motif.

3. **Motif Prediction** Use the profile matrix to identify the most probable motif in each sequence.

4. **Motif Set Update** Update the motif set by replacing one motif with a new one from the corresponding sequence.

5. **Score Calculation** Calculate the score of the motif set using the profile matrix and the consensus sequence.

6. Repeat steps 2 and 3 for a certain number of iterations or until the score converges.

7. Return the motif set with the lowest score.

Since the RMS algorithm initializes randomly, it may take a long time to converge to an optimal solution. In most cases, it fails to reach the global optimum, as the quality of the solution depends on the initial set of motifs. In practice, the algorithm is run multiple times with different initializations to improve the chances of finding the optimal solution.

The algorithm for RMS is as follows:

```
Data: A set of sequences $D$, motif length $k$, number of iterations $N$
Result: A set of motifs $M$
$Motif \leftarrow$ Randomly select k-mers from each sequence in $D$;
$BestMotif \leftarrow Motif$;
for $i \leftarrow 1$ to $N$ do
    $P \leftarrow Profile(Motif)$; $Motif \leftarrow MotifPrediction(D, P)$;
    if $Score(Motif) < Score(BestMotif)$ then
        $BestMotif \leftarrow Motif$;
    end
    else
        return $BestMotif$
    end
end
```

Algorithm 1: Randomized Motif Search

3.2 Gibbs Sampler

Gibbs Sampler is a probabilistic sampling algorithm that refines motif predictions based on the posterior distribution of motif occurrences. It is particularly useful for identifying motifs in a set of sequences with variable lengths and background noise. It iteratively discards one l-mer at a time and replaces it with a new one, based on the probability distribution of the motif occurrences.

The steps of the Gibbs Sampler algorithm are as follows:

1. **Initialization** Randomly select k-mers from each sequence in the dataset. This forms the initial set of motifs.

2. **Sequence Selection** Randomly select one sequence from the dataset.

3. **Motif Profile Construction** Construct a profile matrix from the all but the selected motif, consisting of the frequency of each nucleotide at each position in the motif.

4. **Motif Prediction** Use the profile matrix to identify the most probable motif in the selected sequence.

5. **Motif Set Update** Update the motif set by replacing the selected motif with the most probable one from the corresponding sequence.

6. **Score Calculation** Calculate the score of the motif set using the profile matrix and the consensus sequence.

7. Repeat steps 2 to 5 for a certain number of iterations or until the score converges.

8. Return the motif set with the lowest score.

The algorithm for Gibbs Sampler is as follows:

```
Data: A set of sequences $D$, motif length $k$, number of iterations $N$
Result: A set of motifs $M$
$Motif \leftarrow$ Randomly select k-mers from each sequence in $D$;
$BestMotif \leftarrow Motif$;
for $j \leftarrow 1$ to $N$ do
    $i \leftarrow Random(|D|)$;
    $P \leftarrow Profile(Motif - Motif[i])$;
    $Motif[i] \leftarrow MotifPrediction(D[i], P)$;
    if $Score(Motif) < Score(BestMotif)$ then
        $BestMotif \leftarrow Motif$;
    end
end
return $BestMotif$
```

Algorithm 2: Gibbs Sampler

4 Software

We use two widely used motif discovery tools, MEME and STREME. Both tools are open-source and availble in The MEME Suite [1]. We obtained the source code for the suite and compiled it on our local machine for uniform comparison between the tools.

4.1 Installation

We installed the MEME Suite on our local machine. We used the following commands to install it on Debian-based systems.

```
$ wget https://meme-suite.org/meme/meme-software/5.5.5/meme-5.5.5.tar.gz
$ tar -xvzf meme-5.5.5.tar.gz
$ cd meme-5.5.5
$ ./configure --enable-build-libxml2 --enable-build-libxslt
$ make
$ make test
$ make install
```

4.2 MEME

MEME, or Multiple EM for Motif Elicitation [2], is a widely used motif discovery tool. It uses the Expectation Maximization (EM) algorithm to find motifs in a set of sequences. The tool also provides a statistical significance score for each motif.

Usage

We used the following basic command to run MEME on our dataset.

```
$ meme <input.fasta> -dna -w <width> -text
```

4.3 STREME

STREME, or Sensitive, Thorough, Rapid, Enriched Motif Elicitation [3], is another motif discovery tool that uses the EM algorithm to find motifs in a set of sequences. It extends the capabilities of MEME by incorporating additional features tailored for regulatory motif analysis.

4.3.1 Usage

We used the following basic command to run STREME on our dataset.

```
$ ./streme --p <input.fasta> -nmotifs 1 -minw <min-width>
-maxw <max-width> -thresh <threshold> -dna anr -text
```

4.4 Scripts

We wrote a couple of scripts to automate the process of running MEME and STREME on our dataset. The scripts are written in Bash and are as follows:

4.4.1 memes_run.sh

```
#!/bin/bash

# Directory where output files will be stored
output_dir1="hm03"
output_dir2="yst04r"
output_dir3="yst08r"

# Check if the output directory exists, if not, create it
if [ ! -d "$output_dir1" ]; then
  mkdir -p "$output_dir1"
fi
if [ ! -d "$output_dir2" ]; then
  mkdir -p "$output_dir2"
fi
if [ ! -d "$output_dir3" ]; then
  mkdir -p "$output_dir3"
fi

outfile_counter=10
for w in {8..16}; do
```

```
  ./meme hm03.fasta -dna -w $w -text >
    "${output_dir1}/out${outfile_counter}.txt"
  ((outfile_counter++))
done

outfile_counter=10
for w in {8..16}; do
  ./meme yst04r.fasta -dna -w $w -text >
    "${output_dir2}/out${outfile_counter}.txt"
  ((outfile_counter++))
done

outfile_counter=10
for w in {8..16}; do
  ./meme yst08r.fasta -dna -w $w -text >
    "${output_dir3}/out${outfile_counter}.txt"
  ((outfile_counter++))
done
```

4.4.2 stremes_run.sh

```
#!/bin/bash

# Directory where output files will be stored
output_dir1="hm03"
output_dir2="yst04r"
output_dir3="yst08r"

# Check if the output directory exists, if not, create it
if [ ! -d "$output_dir1" ]; then
  mkdir -p "$output_dir1"
fi
if [ ! -d "$output_dir2" ]; then
  mkdir -p "$output_dir2"
fi
if [ ! -d "$output_dir3" ]; then
  mkdir -p "$output_dir3"
fi

outfile_counter=1
for w in {8..16}; do
  ./streme --p hm03.fasta -nmotifs 1 -minw $w -maxw $w
    -thresh 0.05 -dna anr -text >
        "${output_dir1}/out${outfile_counter}.txt"
  ((outfile_counter++))
done

outfile_counter=1
for w in {8..16}; do
  ./streme --p yst04r.fasta -nmotifs 1 -minw $w -maxw $w
```

```
    -thresh 0.05 -dna anr -text >
    "${output_dir2}/out${outfile_counter}.txt"
  ((outfile_counter++))
done

outfile_counter=1
for w in {8..16}; do
  ./streme --p yst08r.fasta -nmotifs 1 -minw $w -maxw $w
    -thresh 0.05 -dna anr -text >
    "${output_dir3}/out${outfile_counter}.txt"
  ((outfile_counter++))
done
```

5 Results

Here, we summarize the results of our experiments. We list the most probable motif found for each length of k-mer from 8 to 16 for both algorithms and compare them to the results found from the tools. We also recorded the scores for the best motifs from each tool.

5.1 hm03

The following tables contain the motifs with the best score for each length from the `hm03` dataset. The first table contains the results from the algorithms, and the second table contains the results from the tools.

Motif Length	Randomized Motif Search	Gibbs Sampler
8	AAAAGAAA	CAGCCAAT
9	AAAAATCAA	AACTGGTCC
10	TTTGCTCTTC	TCTGGTTCAT
11	GGCTCAGTGCC	AAATGGTTGTGAG
14	AGTGAAAAAAAAAT	CTAGAAGTTTCTAA
15	GAATGAAAAAAAAAT	AAAAAAGAAAGAAAT
16	GAAAACAAAAAAAAAA	TTACTTCAAGAGAACA

Table 1: Most probable motifs found in `hm03` for the algorithms

Motif Length	MEME	E-Value	STREME	Score
8	GGCGGGAA	8.30E+01	GTAAATAA	1.70E-05
9	TTTCTGGCA	1.60E+01	AAGAAAAGA	1.70E-05
10	ACACCCAGAC	3.20E+00	AGGAAGAAAA	1.70E-05
11	CACCCAAACAC	3.60E-01	GAAAAGAAAAA	1.40E-06
12	AAAAAGAAAAAG	4.80E-03	AAAAAAAAAAAT	1.70E-05
13	AAAAGGAAAAAGA	5.20E-03	AAAAAAAAAAATA	1.70E-05
14	ACACCCAGACACCC	4.70E-04	AAAAAAAAAAATAG	1.40E-06
15	TACACCCAGACACCC	1.00E-03	AAAAAAAAAAATAGG	1.40E-06
16	GCAAAGGAGAGAAAGC	2.70E-01	GGAAAAAAAAAAATAG	1.70E-05

Table 2: Most probable motifs found in `hm03` for the tools

5.2 yst04r

The following tables contain the motifs with the best score for each length from the `yst04r` dataset. The first table contains the results from the algorithms, and the second table contains the results from the tools.

Motif Length	Randomized Motif Search	Gibbs Sampler
8	ATTTTTTT	CATTGTAA
9	TTTTTTTTC	TAATCTTTT
10	TTATTTTTCT	CAAAATAAAC
11	AAAAAAAAAAA	TGATTGTGGGA
12	TATTTTTCTTTT	AAAAAAAAAAAA
13	GAAAAAAAAAAAA	CCTTTTTTTTTTA
14	TATTTTTCTTTTTT	TTTTCTTTCTTTCT
15	TATTTTTTTTTTTTT	AAAAAAAAAGAATAA
16	TTTTTTATTCTTTTCT	AAATAAAAAAAAAAAA

Table 3: Most probable motifs found in `yst04r` for the algorithms

Motif Length	MEME	E-Value	STREME	Score
8	GCGGCGGG	1.50E+02	AATTAATT	2.90E-04
9	TTTCTGGCA	3.10E+02	AAATTAATCT	2.90E-04
10	CTGGCATCCA	2.50E+01	AAGGTATATA	2.90E-04
11	CTGGCATCCAC	9.20E+00	ACAAGAGAGAA	2.90E-04
12	CTGGCATCCACT	9.20E+00	AAAAAATTAATG	2.30E-03
13	TTTTCTGGCACCC	5.00E+01	AGAAAAGAAAAAA	2.90E-04
14	ACCCAGACATCTGG	7.90E+01	AGAAAAGAAAAATA	2.90E-04
15	CTTTTCTGGCACACA	1.70E+01	AGAAAAGAAAAAAAA	2.90E-04
16	CCTTTTCTGGCAACCA	9.80E+00	AGAAAAGAAAAAAAAA	2.30E-03

Table 4: Most probable motifs found in `yst04r` for the tools

5.3 yst08r

The following tables contain the motifs with the best score for each length from the `yst08r` dataset. The first table contains the results from the algorithms, and the second table contains the results from the tools.

Motif Length	Randomized Motif Search	Gibbs Sampler
8	TTTTTTTT	AAAAATAT
9	ATTTTTTTT	TTTTTTTTA
10	ATTTTTTTTT	AATTCAATAA
11	ATTTTTTTTTT	AGTTTGTAATA
12	TATTTTTTTTTT	TTATTTTTTTTT
13	TATTTTTTTTTTT	TTTTTTTATTTTT
14	TCTATTTTTTTTTT	AATTTGAATTTTTT
15	TTTTTATTTTTTTCT	ACTACTTTTCGTATT
16	TTTCTTTTTTTTATTT	CCTATTACTTGCATTT

Table 5: Most probable motifs found in `yst08r` for the algorithms

Motif Length	MEME	E-Value	STREME	Score
8	GGCGGGAA	8.30E+01	GTAAATAA	1.70E-05
9	TTTCTGGCA	1.60E+01	AAGAAAAGA	1.70E-05
10	ACACCCAGAC	3.20E+00	AGGAAGAAAA	1.70E-05
11	CACCCAAACAC	3.60E-01	GAAAAGAAAAA	1.40E-06
12	AAAAAGAAAAAG	4.80E-03	AAAAAAAAAAAT	1.70E-05
13	AAAAGGAAAAAGA	5.20E-03	AAAAAAAAAAATA	1.70E-05
14	ACACCCAGACACCC	4.70E-04	AAAAAAAAAAATAG	1.40E-06
15	TACACCCAGACACCC	1.00E-03	AAAAAAAAAAATAGG	1.40E-06
16	GCAAAGGAGAGAAAGC	2.70E-01	GGAAAAAAAAAAATAG	1.70E-05

Table 6: Most probable motifs found in `yst08r` for the tools

6 Conclusion

Since we used Hamming distance as our scoring function, this ended up causing a bias against larger motifs, as larger sequences are more prone to higher Hamming distances in general. Moreover, since MEME starts with a random initialization and changes multiple motifs per iteration, it generally takes longer to reach an optimal set of motifs, thus STREME usually has a faster runtime that MEME for all three datasets. Furthermore, improvements in STREME nakes it more tolerant towards noise and variations of sequence in motifs, making it a better choice for real-life datasets that are prone to error. Neither of our algorithms produced consistent results however, probably due to a smaller number of repetitions. Further study is required using larger datasets and different heuristic functions to provide a more comprehensive comparison of the algorithms and tools.

References

[1] The MEME Suite. https://meme-suite.org/meme/.

[2] T L Bailey and C Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc Int Conf Intell Syst Mol Biol*, 2:28–36, 1994.

[3] Timothy L Bailey. STREME: accurate and versatile sequence motif discovery. *Bioinformatics*, 37(18):2834–2840, 03 2021.