EMD Algorithm: A journey to Replication and Improvement of the algorithm

Dafni Tsolisou[1]
daphnetsolissou@gmail.com
Athens, Greece

Spyridon Alvanakis-Apostolou
s.alvanakis@gmail.com
Athens, Greece

ABSTRACT

Motif discovery algorithms play a vital role in identifying significant patterns in biological sequences. This paper presents the implementation and evaluation of the Ensemble Motif Discovery (EMD) algorithm, as described in the original EMD paper. Three experiments were conducted using the ECRDB61B-200 dataset, and the results were compared with the reported findings. The accuracy scores obtained were lower than those reported in the original paper. The study concludes that multiple runs and the ensemble approach enhance the performance of motif discovery algorithms. The findings also highlight the potential of the weighting strategy using the STREME algorithm, which demonstrated improved results and holds promise for further optimization.

KEYWORDS

Regulatory Motif Discovery, Motif Positions, Ensemble Method, E. Coli Genome, Bioinformatics

1 INTRODUCTION

Sequence motifs are short similar recurring patterns of nucleotides, found in the upstream regions of genes in eukaryotic cells, that have some biological meaning. Correctly identifying regulatory motifs and binding sites is an important task since it has the potential to unravel the underlying mechanisms of diseases and biological processes.

Although it has been a prominent focus in bioinformatics reasearch, it still remains a challenging task due to the inherent characteristics of sequence motifs. In general, motifs are consistent in size and have a tendency of being repeated and conserved within genomes. However, the fact that they are relatively small in length, typically spanning around 6-12 base pairs, poses a challenge in discovering them within intergenic regions, which are considerably lengthy and highly variable. [5] This makes for a low noise-to-ratio problem.

Numerous motif discovery algorithms have been developed to tackle this challenge based on statistical and probabilistic approaches. Nonetheless, the overall accuracy of such algorithms still remains relatively low, leaving room for further research and improvement. [7]

One way of achieving that is by adopting an ensemble strategy, which harnesses the collective power of multiple existing motif discovery algorithms. By combining the solutions generated by these algorithms, an ensemble approach aims to produce a superior solution through majority consensus.

In this study, we present an endeavor to replicate and improve upon an ensemble motif discovery (EMD) algorithm. The EMD algorithm implementation that is presented incorporates five well-known component algorithms and demonstrates the potential for enhancing motif discovery outcomes through the synergy of their combined efforts.

In this study, our objective is to replicate the results of the aforementioned EMD algorithm in an effort to validate its results and potentially improve them. Instead of relying solely on probabilistic algorithms as in the original algorithm, we incorporate an additional word-based method that assigns different weights to the predictions. By applying this novel addition, we strive to achieve better prediction results compared to the original implementation.

[1]Both authors contributed equally to this research.

Our approach serves as an independent effort to build upon existing research, demonstrating the potential for alternative strategies to improve the accuracy of motif discovery algorithms. By exploring different approaches to ensemble-based motif discovery and incorporating innovative weighting schemes, we aim to provide valuable insights and advancements in this important field of bioinformatics.

2 BACKGROUND AND RELATED WORK

The problem of motif discovery in intergenic regions can be considered computationally as extracting signal from noise. In general, the problem can be formulated as having a set of sequences in which exists an unknown pattern or a motif that is repeated multiple times and needs to be found.

There are numerous motif discovery algorithms available in the literature, broadly classified into two main types: enumeration-based and probabilistic-based approaches [5]. Enumeration techniques aim to identify consensus motifs by exhaustively exploring the search space through word enumeration and computing word similarities. Although these techniques can yield the global optimum solution, they often come with substantial computational and memory costs due to the exhaustive nature of the search.

On the other hand, probabilistic approaches construct statistical models called motif matrices. These matrices reveal a distribution of the four bases for each position in a sequence and are used to predict the most likely locations of motifs in sequence. For motifs of width W, a matrix of size $(Wx4)$ is built, where each column represents the distribution of one of the four nucleotides at the various positions in the motif represented by each row. [2]

One way to improve the probabilistic approach is by involving advanced motif models capable of capturing the characteristics of sequence patterns, such as position-dependence information. Hidden Markov models and local pairwise nucleotide frequency analysis have been proposed as alternatives to the conventional position-specific scoring matrix. Additionally, employing more sophisticated search algorithms in the sequence space, such as Expectation Maximization or Gibbs sampling, has been investigated. Another direction for improvement involves incorporating additional information, such as phylogenetic trees, homologous sequences, comparative genomics, and gene expression data, to enhance the specificity of motif discovery.

The Gibbs' sampling algorithm [10] is a famous stochastic method that leverages Markov chain Monte Carlo (MCMC) techniques to generate a sequence of samples from the motif instance space. Many motif discovery methods extend the functionality of this algorithm in order to increase sensitivity.

An ensemble method for motif discovery can employ different types of algorithms in order to leverage the predictive power of each one and reach a consensus solution of superior accuracy. The EMD algorithm that is described in this paper [7] by Hu and Kihara claims that its success over single prediction methods stems from the systematic combination of five popular algorithms: AlignACE, Bioprospector, MDScan, MEME and MotifSampler.

These algorithms, known for their probabilistic nature, contribute diverse perspectives to the motif discovery process. Notably, MotifSampler and BioProspector are non-deterministic algorithms, adding an element of randomness to their predictions.

AlignACE [14] utilizes a combinatorial approach to identify conserved DNA motifs within a set of input sequences. It is an extension of the Gibbs Motif Sampling algorithm, that scans non-coding nucleic acid sequences at high resolution for motifs which occur with non-random frequency. It implements two proposed scoring methods, MAP and group specificity, to evaluate the alignment significance by their frequency of occurrence.

Bioprospector [11] is a widely used algorithm in computational biology, which employs the Gibbs sampling strategy for motif detection. Bioprospector, improves upon the Gibbs strategy by utilizing zero to third-order Markov background models, and a motif score distribution estimated by a Monte Carlo

method. It is also designed to discover gapped motifs and palindromic patterns.

MDScan [12] combines the advantages of word-enumeration and position specific weight matrix updating, while being faster and nearly as sensitive as Bioprospector. It is a strictly deterministic greedy algorithm, designed to avoid local minima and always give the globally best conserved motif.

MEME [4] is a deterministic algorithm that extends the expectation maximization (EM) technique for identifying motifs in unaligned biopolymer sequences. It discovers motifs in sequences for which neither the number of motifs nor the number of occurences of each motif is known. It starts from an initial motif and then uses expectation and maximization steps to improve upon it iteratively until a stop condition is met.

MotifSampler [15] is also based on Gibbs' sampling but includes two notable modifications. The first is the incorporation of a probability distribution to estimate the number of copies of a motif in each sequence. The second is the inclusion of a high-order Markov background model.

For this implementation an additional algorithm was incorporated and is be briefly described. STREME [3] is a word-based algorithm that belongs to the MEME Suite. It is fast, suitable for large sequence datasets and more sensitive than MEME. It uses a suffix tree and provides accurate estimates of the statistical significance of the motifs it discovers. Moreover, it utilizes Fisher's exact test if a control sequence of the same length distribution as the primary sequence is given, and the Binomial test otherwise.

3 METHOD

3.1 Benchmark Datasets

The original EMD paper provides three types of benchmark data sets derived from the binding site (motif) information of Escherichia coli K-12, obtained from the database RegulonDB [16]. Additionally, the entire genome of E. coli, sourced from the KEGG database [8] was used to generate these data sets, as well as information about starting and ending positions of genes.

Prior to motif discovery analysis, all data sets underwent preprocessing, including assembly and cleaning, to ensure optimal suitability for the task at hand. During the assembling step the binding site records in RegulonDB are organized in groups which bind to the same transcription factor, called motif groups. Then each motif group is cleaned in order to remove records that do not have positional information on the genome, are duplicates, or differ with other binding site records only by a <5 nt shift. Finally, motif groups with only one sequence were removed. This data set was used as the basis for building the benchmarks.

Each benchmark type has different characteristics which are briefly described:

- Type A: this data set was generated from the intergenic regions of E. coli. The extraction was done by aligning each known binding site of a motif group to the genome, locating the adjacent genes to the binding site and extracting the intergenic region to generate one input sequence. The final set contains 62 motif groups, each comprising on average 12 sequences of 300 nt width and 1.85 sites per sequence. Each site has an average width of. 22.83
- Type B: this data set is extracted similarly as Type A but is modified to contain symmetric margins of fixed length on both sides. Depending on the margin size a sequence may contain more than one motifs. This concept of sequence extraction is illustrated in Figure 1
- Type C: this data set has also symmetric margins like B, but they are generated artificially around known binding sites.

For this study only the Type B data set of margin size 200 was used during the development process and for all experiments. The given data set includes

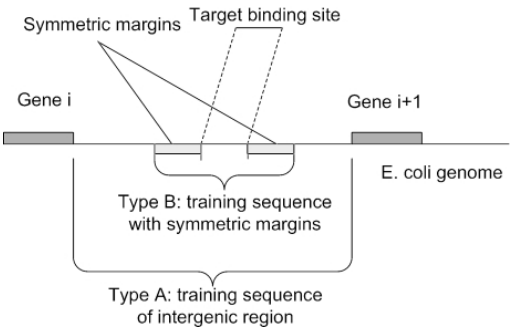


Figure 1: Two types of generated input sequences from the E. coli genome using known binding sites and gene positions. [6]

61 motif groups, where each group does not contain different motifs multiple times and is referred as ECRDB61B-200.

The original EMD paper provides results for different algorithm combinations, tested on this particular data set. Hence, we chose to focus on this benchmark set in order to compare performance.

3.2 Algorithm Setup

In our research, we standardized the desired width of the predicted motifs to 15 for all the algorithms used. Additionally, we generated a background file using a transition matrix with 3rd order Markov chain probabilities. This background file is crucial for the probabilistic algorithms to make accurate predictions. It is important to note that each algorithm requires a specific format of the background file, meaning that a single file cannot be used across all the algorithms. Also, to ensure the reliability of our research, we executed each algorithm a total of 10 times.

However, it is worth mentioning that the STREME algorithm differs from the others in this regard. Unlike the probabilistic algorithms, STREME is a word-based algorithm that relies on frequencies rather than probabilities to make predictions. Therefore, STREME does not require a background file for its operation.

3.2.1 MEME & MotifSampler. The MEME (ME) and MotifSampler (MS) algorithms are considered to be more user-friendly compared to other probabilistic algorithms. During our research, we discovered numerous online (we used http://rsat.sb-roscoff.fr/create-background-model_form.cgi) resources that facilitated the creation of the background file required to run these algorithms effectively. While both algorithms contributed to the ensemble approach, it is worth noting that MotifSampler, being a non-deterministic algorithm, exhibited lower time performance. Specifically, running MotifSampler with the mentioned parameters took approximately 4-5 hours. Despite the longer execution time, we deemed it worthwhile due to the valuable insights it provided.

3.2.2 MDScan & Bioprospector. MDScan (MD) and Bioprospector (BP), both algorithms share a similar input structure. They require a FASTA file containing the sequences for prediction and a corresponding background model. However, when utilizing supplementary algorithms from the packages, we encountered difficulties in handling the relatively small size of the background sequences used for generating the background file. The supplementary code imposed a size limit on the input background sequence, which proved to be too small even for our already small-sized E.Coli K-12 background sequence. Moreover, when working with smaller sequences, we observed that the background file provided negative numbers. Further investigation revealed that these given numbers were logarithmic probabilities, with specific rows indicating the probabilities of the Markov chain order. Additionally, we found the letter order within the provided probabilities, enabling us to develop our own code for calculating the background file using the complete E.Coli K-12 genome. After resolving the background file issue, we proceeded to run the algorithms, with MDScan demonstrating the fastest performance among all tested algorithms.

3.2.3 AlignAce. This algorithm's software is available for download through this url. The linux version was downloaded on a machine running Kubuntu 22.04 and extracted in a local directory. Running was

a failue as a explained in the next paragraph and this is why the STREME algorithm was selected as a replacement.

Initial efforts to run the software failed with this missing library error: "./AlignACE: error while loading shared libraries: libstdc++-libc6.1-1.so.2: cannot open shared object file: No such file or directory". As suggested in this stack overflow answer, a customlibs directory was created along with a soft link between the not-matching library filename and the corresponding libstdc++.so.6.0.30 system library. This caused the following error: "./AlignACE: symbol lookup error: ./AlignACE: undefined symbol: cerr", which means that C is searching for a missing function. To solve this different paths like lib32, were tested in order to find the one matching the same architecture at the time AlignAce was developed. Unfortunately, the error persisted and since we ran out of ideas on how to make it work we abandoned it completely.

3.3 The Pipeline

The original pipeline can be seen in Figure 2, which includes five distinct steps: Collection, Grouping, Voting, Smoothing, Extracting. The description of each step as provided by the paper was our main source for replicating the algorithm using Python. It was our intention to remain as faithful as possible to the original algorithm, making as less assumptions as possible.

3.3.1 Collection. The steps of the collection phase as described in the paper are the following:

(1) Run each algorithm Ai R times against an input dataset of N sequences.
(2) Report K scoring motif, one for each run. This results in R*K total predictions per Ai.
    (a) If Ai predicts more than K keep top K predictions.
    (b) If Ai predicts less than K keep all predictions.
    (c) If more than 1 sites or no sites are predicted then more or less than R*K sites are returned.

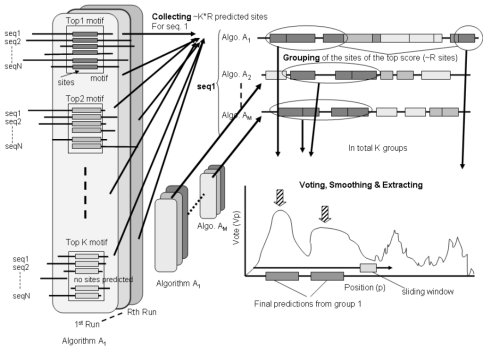


Figure 2: Overview of the EMD algorithm as described in the original pipeline. [7]

(3) Collect all predicted sites for each combination of input sequence Si and Ai.

Running each component algorithm multiple times on an input data set produces multiple output files. Each output file corresponds to one motif group of the input data set and contains a certain number of discovered motifs, usually five. For each discovered motif, all the positions of the sites where it was found in each sequence belonging to the corresponding motif group is included, along with a score. Since one or more sites are predicted for each sequence, we can deduct that all of the sites predicted for an input sequence set (motif group) form a motif.

For our implementation, we run the algorithms R=10 times on the entire benchmark data set B, with margin size 200, using Bash scripts. The results were stored as txt files on separate directories for each algorithm. The top K=5 scoring motifs were kept, as was done in the initial paper also.

To extract the appropriate information from each txt file regular expressions [1] was employed. A regular expression is a sequence of characters that define a search pattern. The re module which is readily available in Python, provides regular expression matching operations. Extensive use of this module was done in order to parse motif and site information.

Different search patterns for each algorithm's output file were created. Each pattern was designed to extract similar information in order to produce normalized results that can be easily analyzed in subsequent steps. What was important to store from each text was the width and score of each discovered motif, all sites per motif and their starting positions. Additionally, for each site the corresponding motif group name and a unique id that indicates the sequence from which a site was derived, was also stored. The final data structure that was used to store motif information was a Pandas Dataframe. [13]

The sequence id was the identifier which was used to perform the ensemble steps that are described subsequently. It was the connection between predicted motifs and target motifs used for evaluation.

Because the STREME output file did not contain any information about sites in specific sequences and their positions, it was impossible to normalize its output in the same way as for the other algorithms. This is why this algorithm was not utilized in the same way as the rest.

3.3.2 Grouping. The steps of the grouping phase as described in the paper are the following:

(1) Sort all predicted sites in an input sequence Si by a certain algorithm Ai by the algorithm's major statistical score.
(2) Divide predicted sites into K groups by the sorted score, with each of the groups having an equal number of predicted sites. Usually each group has R sites.
(3) Join together the groups of the same score rank across the results by M different algorithms. Usually K groups.
(4) Return all predicted sites for K groups

The primary objective of the grouping phase was to organize the data collected during the procedure based on sequence IDs and their corresponding scores. Initially, scores were assigned to different score groups, ranging from 1 to 5, which indicated the quality of the predictions for each algorithm. The score group 5 included the highest scores, while score group 1 represented the lowest scores. Subsequently, the predictions were grouped using their scoring group, file name, and sequence ID. Although the sequence IDs are unique within each file, grouping by file was performed to facilitate later tracking of the predictions specific to each file. Finally, the total number of groups was found to be 2232.

3.3.3 Voting. The steps of the voting phase as described in the paper can be seen below.

(1) For each of the K predicted site groups in a sequence Si, place all the predicted sites in a group on the sequence.
(2) For each position p in a sequence, count the number of times the position p is included, or votes for the position p, Vp, in the predicted sites.

In Figure 3 a schematic representation of the voting can be seen.

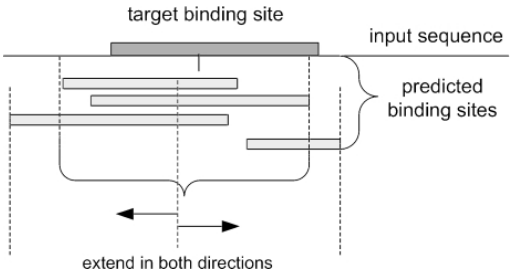


Figure 3: Top predictions from multiple runs are aligned together to count the frequency in each position. [6]

At the voting phase for each of the 2232 groups, we identified the positions where each predicted motif exists. For each position where a motif was found, we incremented the vote count by 1. In cases where there were areas with multiple predicted motifs, we assigned votes to those areas based on the number of motifs associated with them. At Figure 3 illustrates this process, showcasing areas with 1, 2, or 3 votes depending on the corresponding motifs.

Figure 4a provides an example image illustrating the vote distribution for Ada_1_209298-209425-forward, Ada_2_209298-209425-forward, and Ada_1_2308475-2308502-reverse. Notably, in Ada_1_209298-209425-forward, the highest vote concentration appears around position 270. However, a challenge arises in the voting phase due to the potential lack of a unique global maximum position. This occurs when multiple areas receive the same vote count. To address this issue, we implemented the smoothing phase, as described later.

3.3.4 Smoothing. The steps of the smoothing phase as described in the paper are the following:

(1) Scan the sequence left to right and sum the votes Vp in a window of width Ws, which should be hhalf the length of motif width W.
(2) Place the sum Vps in the center position p of the window. (If Ws is even place Vps at position q = (Ws/2)+1)

The smoothing phase solves the challenge of determining the global maximum in the voting phase. By applying the sliding window technique with a window size equal to half the width of the predicted motif, we successfully smooth the voting plots. Figure 4 clearly illustrates the distinction between the smoothed plot and the original voting plot. The smoothed plot significantly enhances the visibility of the global maximum point, allowing us to identify the position with the highest frequency of predicted motifs. This position is then selected as the final predicted position with the most common occurrence of predicted motifs.

3.3.5 Extracting. The steps of the extracting phase as described in the paper are the following:

(1) Pick up the top local peak (or the top Bi local peaks) in the smoothed voting curve, Vps.
(2) Place a window of the length W as the final prediction of the site, with the center of the window positioned at the peak.

During the extraction phase, the main objective was to retain the predicted areas obtained from the smoothing phase. To identify these predicted areas, we initially identified the position with the highest smoothed votes. We then considered this position as the median position of the predicted motif. To obtain a predicted area with a width of 15, which aligns with the desired predicted motif width, we added and subtracted 7 from the median position. Additionally, we preserved the sites corresponding to the predicted areas to facilitate the subsequent weighting part of our research.

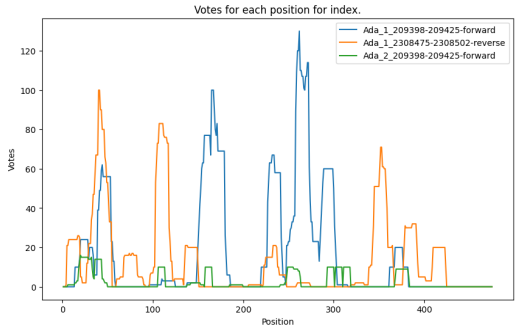
3.4 Weighting Policy

The STREME algorithm, being a word-based algorithm, does not provide the positions of the predicted sites. To incorporate STREME into our analysis, we opted to assign different weights to the predicted sites of the probabilistic algorithms based on the STREME predictions. These weights were determined by comparing the nucleotides at corresponding positions between the motifs. For instance, if a probabilistic algorithm predicts a site as AAAATTTT, and STREME predicts it as AAGGTTTT, there are two differences at the 3rd and 4th positions. In this case, the motif AAAATTTT from the probabilistic algorithm would receive a higher weight than AAAAATTT, which has three differences from the word-based motif. To establish a threshold for similarity, we set at most eight differences as the cutoff. Since there was limited existing literature on assigning weights to predicted motifs, we conducted experiments using different weighting sets, utilizing various ranges and scalings, to assess the impact of the weights on the final results.

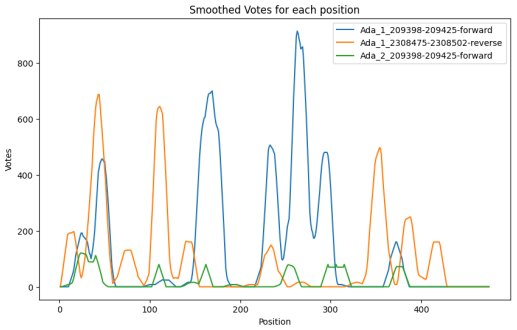
3.5 Evaluation Measures

In order to count the performance of the ensemble algorithm two types of evaluation measures are used in the original paper: a Nucleotide Level Accuracy and a Site Level Accuracy. To make the notion of accuracy, sensitivity and specificity more apparent in the context of motif discovery Figure 5 is provided.

To implement these measures in Python the overlap between a predicted site and a target site was necessary. Known target sites were provided by the supplementary material of the paper and included



(a) Example of Voting results



(b) Example of Smoothing results

Figure 4: The Figures 4a 4b illustrates the results obtained from the Ada file, encompassing scoring groups 1 and 2, along with the sequence IDs 209298-209425-forward and 2308475-2308502-reverse. As depicted in the voting figure, it is evident that there are edges and local maximum points that do not precisely indicate the true maximum point of the voting plot. To overcome this, a smoothing technique was employed to enhance the clarity of the voting plot. After smoothing the voting plot, a notable improvement can be observed in extracting the optimal prediction, as the maximum point became more distinct and discernible.

sequence ids. These ids were used to match our predictions with the ground truth motifs.

The starting and ending positions of the target sites were found by simple string manipulation. The positions of the predicted sites were already available from the extraction phase of the pipeline. Using simple algebraic rules we were able to calculate the overlap between those two ranges.

3.5.1 Nucleotide Level Accuracy. The nucleotide level accuracy is calculated between each target site and a corresponding predicted site for a sequence. It includes the performance coefficient (nPC), the sensitivity (nSn) and the specificity (nSp) metrics, which are calculated position-wise.

To calculate these values the metrics of true positive (nTP), true negative (nTN), false positive (nFP) and false negative (nFN) need to be defined:

- nTP: the number of target binding site positions predicted as binding site positions.
- nTN: the number of non-target binding site positions predicted as non-binding site positions.
- nFP: the number of non-target binding site positions predicted as binding site positions.
- nFN: the number of target binding site positions predicted as non-binding site positions.

Then nSn, nSp and nPC are defined as always:

$$nSn = \frac{nTP}{nTP + nFN} \tag{1}$$

$$nSp = \frac{nTP}{nTP + nFP} \tag{2}$$

$$nPC = \frac{nTP}{nTP + nFP + nFN} \tag{3}$$

In case of no overlap between predicted and target sites, the nSn, nSp and nPC were set to zero.

The final accuracy score is the average over all binding site pairs in a sequence, followed by averaging over all sequences in a motif groups, then averaged over all the motif groups.

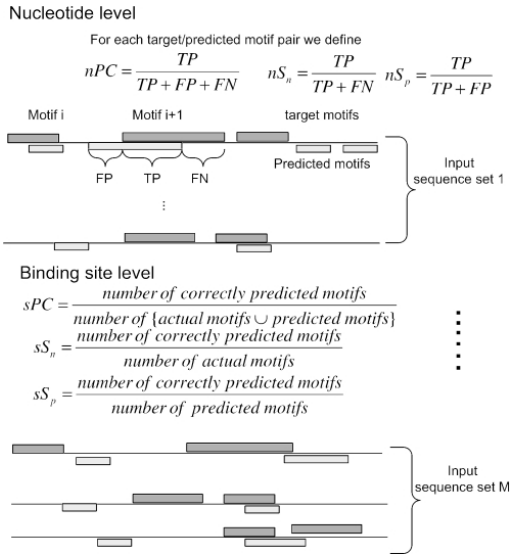


Figure 5: Measures of prediction accuracy at the nucleotide and site levels. [6]

3.5.2 Site Level Accuracy. The site level accuracy include the site level performance coefficient (sPC), the sensitivity(sSn) and the specificity (sSp). It indicates if predicted sites overlap with true sites by one or more nucleotides.

To calculate these values the metrics of true positive (sTP), false positive (sFP) and false negative (sFN) need to be defined:

- sTP: the number of predicted binding sites which overlaps with the true binding sites by at least 1 nt.
- sFP: the number of predicted binding sites which have no overlaps with the true binding sites.
- sFN: the number of true binding sites that have no overlaps with any predicted binding sites.

Then sSn, sSp and sPC are defined as always:

$$sSn = \frac{sTP}{sTP + sFN} \tag{4}$$

$$sSp = \frac{sTP}{sTP + sFP} \tag{5}$$

$$sPC = \frac{sTP}{sTP + sFP + sFN} \tag{6}$$

The site level accuracy score for an input sequence set is the average of the score over all sequences. The site level accuracy score of the entire benchmark data set is the average of the scores for all input sequence sets.

4 EVALUATION

To evaluate our implementation and compare it with the reported results of the original EMD paper three experiments were conducted. All experiments used the output files of 10 runs of the component algorithms on ECRDB61B-200 data set.

4.1 Experiment 1

For this experiment the motif results of all MEME, Bioprospector, MDscan and MotifSampler were used throughout the EMD pipeline. No weighting scheme using STREME was employed. The final accuracy scores can be seen in Figures 6 and 7.

Compared to the performance results of the original paper which claim nPC scores of a little over 0.2 for 10 runs and 0.254 for 20 runs, our results are lower in terms of accuracy.

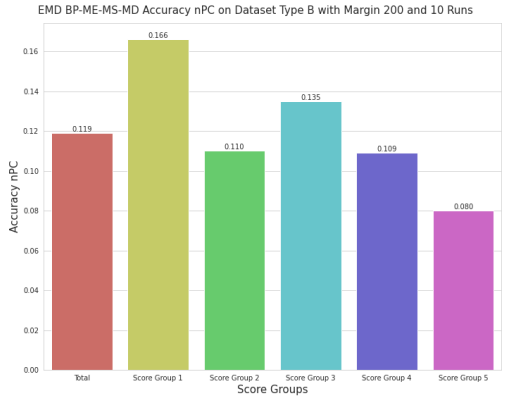


Figure 6: Prediction accuracy nPC at the nucleotide level for experiment 1 total and per score group.

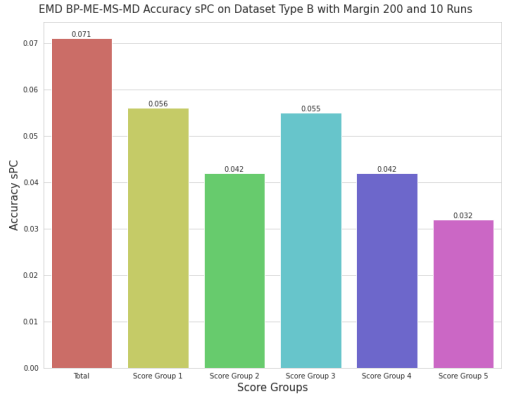


Figure 7: Prediction accuracy sPC at the site level for experiment 1 total and per score group.

4.2 Experiment 2

For this experiment all possible combinations of the four component algorithms are tested and compared in terms of all accuracy scores. The initial recreated pipeline is used to get predictions without usage of the weighting scheme. The complete results can be seen in Table 1.

All these nPC values can be compared with the corresponding scores reported in the original paper for different combinations of component algorithms for 20 runs tested on ECRDB61B-200 data set. Since our results do not include AlignAce and were done for 10 runs instead of 20 we can only draw superficial conclusions.

We noticed that our scores are quite low and somewhat different than those reported in the EMD paper. Instead of having the MDscan as the best scoring algorithm when used single or in combination with other algorithms, we have the MotifSampler.

Table 1: Comparison of prediction accuracy scores of EMD algorithms with different combinations of component algorithms

Algorithms	nPC	nSp	nSn	sPC	sSp	sSn
BP	0.0814	0.174	0.088	0.044	0.068	0.091
ME	0.058	0.111	0.064	0.034	0.046	0.098
MS	0.171	0.266	0.177	0.080	0.097	0.193
MD	0.110	0.197	0.116	0.044	0.073	0.082
BP-ME	0.069	0.144	0.074	0.048	0.057	0.153
BP-MD	0.098	0.195	0.104	0.05	0.073	0.108
MS-ME	0.115	0.195	0.123	0.067	0.075	0.211
BP-MS	0.143	0.24	0.15	0.078	0.089	0.216
MS-MD	0.16	0.256	0.167	0.081	0.093	0.215
ME-MD	0.079	0.146	0.085	0.046	0.056	0.145
MS-MD-ME	0.123	0.206	0.129	0.069	0.077	0.226
BP-MD-ME	0.081	0.159	0.085	0.051	0.061	0.164
MS-MD-BP	0.146	0.245	0.153	0.079	0.09	0.219
MS-BP-ME	0.114	0.203	0.12	0.069	0.077	0.228
MS-MD-BP-ME	0.119	0.21	0.125	0.071	0.078	0.231

4.3 Experiment 3

For the third experiment, we incorporated weights obtained from the STREME algorithm. Due to the

absence of relevant literature, we conducted experiments with different weighting sets2. Motif similarity was set as a criterion for weighting, with a minimum requirement of 7 similarities (or at most 8 differences). This decision was based on the rarity of two motifs having 7 similar nucleotides at the same position of a site. The weights were applied to predictions made by both the STREME and EMD algorithms, specifically within the same files. The weighting sets had varying ranges and scaling factors to evaluate their impact on the results.

Table 2: Weight Sets

Weight Set 1	Weight Set 2	Weight Set 3
0: 2	0: 3	0: 10
1: 1.8	1: 2.5	1: 9
2: 1.6	2: 2	2: 8
3: 1.4	3: 1.5	3: 7
4: 1.2	4: 1	4: 6
5: 1	5: 0.5	5: 5
6: 0.8	6: 0.3	6: 4
7: 0.6	7: 0.2	7: 3
8: 0.2	8: 0.1	8: 1

The weighting system was applied to the extracted predictions for the 2232 groups. Figure 8 illustrates that the STREME algorithm discovered similarities with nearly half of the predicted motifs. Most of these similarities occurred with 8 differences, although there were instances of complete similarity without any differences.

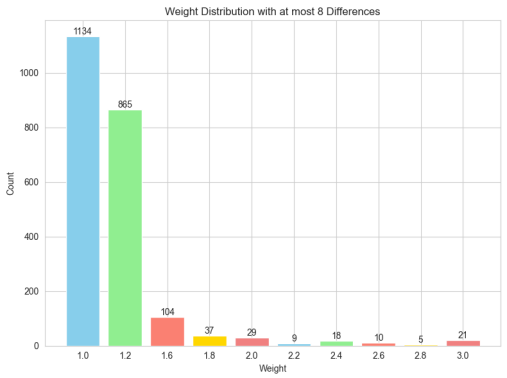


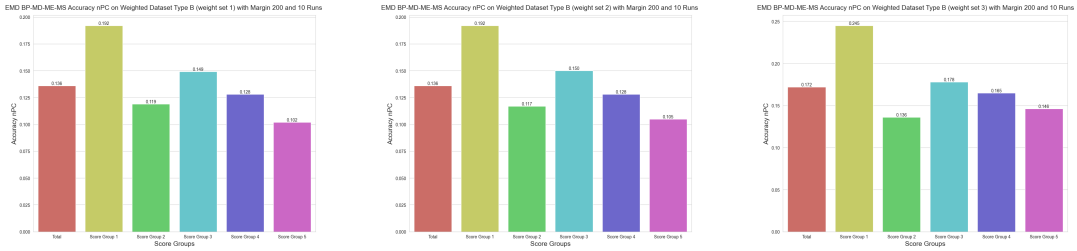
Figure 8: The figure illustrates the distribution of weights for the 2232 groups, specifically for the first weight set. The x-axis represents the assigned weights at the predicted positions, while the y-axis corresponds to the number of predicted positions that received a weight.

Applying the same evaluation metrics as in experiments 1 and 2, we observed a significant improvement. Figure 9 demonstrates that the improvement increased with higher weights in the weight sets. Notably, weight set 1 and weight set 2 had minor differences, while weight set 3 exhibited a substantial difference in the final nPC score, rising from 0.119 without weights to 0.172 when using weight set 3.

Overall, the weights assigned greater importance to specific positions, leading to more accurate predictions compared to those without weights.

4.4 Comments

Based on the results obtained from all the experiments conducted, it is evident that the number of runs plays a crucial role in determining the final outcomes. Significant differences were observed between the results obtained from 10 runs of our study and 20 runs from the researchers study, highlighting the importance of multiple runs. This difference can be attributed to the fact that some algorithms are deterministic, while others are not, resulting in varying results across different runs.



(a) Weight set 1 results **(b) Weight set 2 results** **(c) Weight set 3 results**

Figure 9: Comparison of accuracy scores nPC when applied the weighting scheme using 3 different weighting sets.

Moreover, the synergistic effect of combining multiple algorithms tends to enhance the final nSP score. This observation indicates that ensemble algorithms can produce better results compared to the usage of individual algorithms alone.

Additionally, the weighting method employed in the research has shown promising results. The synergy between probabilistic and word-based algorithms in the weighting method has proven effective. However, there is a drawback in terms of losing information for predicted positions that the word-based algorithm fails to predict. To address this limitation, the incorporation of more algorithms and a better-balanced weighting set could prove beneficial for an ensemble algorithm.

5 CONCLUSIONS

In conclusion, this study aimed to replicate and evaluate the Ensemble Motif Discovery (EMD) algorithm by comparing its performance against the reported results in the original paper. Three experiments were conducted using the ECRDB61B-200 dataset, focusing on different aspects of the EMD pipeline.

Experiment 1 utilized the motif results from four component algorithms (MEME, Bioprospector, MDscan, and MotifSampler) without employing a weighting scheme. The obtained accuracy scores were lower than those reported in the original paper, suggesting a discrepancy in performance.

Experiment 2 explored various combinations of the component algorithms to assess their impact on accuracy. Interestingly, our results differed from the original paper, with MotifSampler outperforming MDscan as the best scoring algorithm, indicating potential variations in algorithm performance.

Experiment 3 introduced the incorporation of weights obtained from the STREME algorithm, using motif similarity as the weighting criterion. Different weighting sets were evaluated, and it was observed that the usage of weights led to significant improvements in prediction accuracy, especially with higher weights. This highlights the effectiveness of the STREME algorithm in enhancing the overall performance of the EMD algorithm.

The results suggest that multiple runs and the synergistic combination of component algorithms contribute to improved accuracy. Moreover, the integration of the STREME algorithm for weighting demonstrates the potential for further optimization in ensemble motif discovery algorithms.

REFERENCES

[1] Alfred V Aho. 1991. Algorithms for finding patterns in strings, Handbook of theoretical computer science (vol. A): algorithms and complexity.

[2] Mohammad Al Bataineh, Zouhair Al-qudah, and Awad Al-Zaben. 2016. Iterative sequential Monte Carlo algorithm for motif discovery. *IET Signal Processing* 10, 5 (2016), 504–513.

[3] Timothy L Bailey. 2021. STREME: accurate and versatile sequence motif discovery. *Bioinformatics* 37, 18 (2021), 2834–2840.
[4] Timothy L Bailey and Charles Elkan. 1995. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine learning* 21 (1995), 51–80.
[5] Fatma A Hashim, Mai S Mabrouk, and Walid Al-Atabany. 2019. Review of different sequence motif finding algorithms. *Avicenna journal of medical biotechnology* 11, 2 (2019), 130.
[6] Jianjun Hu, Bin Li, and Daisuke Kihara. 2005. Limitations and potentials of current motif discovery algorithms. *Nucleic acids research* 33, 15 (2005), 4899–4913.
[7] Jianjun Hu, Yifeng D Yang, and Daisuke Kihara. 2006. EMD: an ensemble algorithm for discovering regulatory motifs in DNA sequences. *BMC bioinformatics* 7 (2006), 1–13.
[8] Minoru Kanehisa, Susumu Goto, Shuichi Kawashima, and Akihiro Nakaya. 2002. The KEGG databases at GenomeNet. *Nucleic acids research* 30, 1 (2002), 42–46.
[9] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. 2016. Jupyter Notebooks – a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt (Eds.). IOS Press, 87 – 90.
[10] Charles E Lawrence, Stephen F Altschul, Mark S Boguski, Jun S Liu, Andrew F Neuwald, and John C Wootton. 1993. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *science* 262, 5131 (1993), 208–214.
[11] Xiaole Liu, Douglas L Brutlag, and Jun S Liu. 2000. BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. In *Biocomputing 2001*. World Scientific, 127–138.
[12] X Shirley Liu, Douglas L Brutlag, and Jun S Liu. 2002. An algorithm for finding protein–DNA binding sites with applications to chromatin-immunoprecipitation microarray experiments. *Nature biotechnology* 20, 8 (2002), 835–839.
[13] The pandas development team. 2020. *pandas-dev/pandas: Pandas*. https://doi.org/10.5281/zenodo.3509134
[14] Frederick P Roth, Jason D Hughes, Preston W Estep, and George M Church. 1998. Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nature biotechnology* 16, 10 (1998), 939–945.
[15] Gert Thijs, Kathleen Marchal, Magali Lescot, Stephane Rombauts, Bart De Moor, Pierre Rouzé, and Yves Moreau. 2001. A Gibbs sampling method to detect over-represented motifs in the upstream regions of co-expressed genes. In *Proceedings of the fifth annual international conference on Computational biology*. 305–312.
[16] Víctor H Tierrafría, Claire Rioualen, Heladia Salgado, Paloma Lara, Socorro Gama-Castro, Patrick Lally, Laura Gómez-Romero, Pablo Peña-Loredo, Andrés G López-Almazo, Gabriel Alarcón-Carranza, et al. 2022. RegulonDB 11.0: Comprehensive high-throughput datasets on transcriptional regulation in Escherichia coli K-12. *Microbial Genomics* 8, 5 (2022).

A SUBMITTED SOURCE CODE

This section is dedicated to the explanation of the submitted source code and instructions for running it. All results should be reproducible. The txt outputs of each component algorithm after running them 10 times on the ECRDB61B-200 is provided in order to run the EMD pipeline. Additionally, the original benchmark data set and a file containing knowing sites of E. coli, used as ground truth for evaluation are accompanying the code. A txt file with all the bash commands we used to run the algorithms on the ECRDB61B-200 is also provided.

Each phase of the pipeline takes up a dedicated python script and the final orchestration of the pipeline is done using a Jupyter Notebook[9] called Run_EMD_Pipeline.ipynb. At the end of each phase, one or multiple csv files are produced to store the results data structures, thus making all steps distinct and minimizing the need to rerun the entire pipeline.

A directory named Results containing the outputs of all algorithms should be found in the current directory of the source code, as well as a directory named CSVs to store all results files of the pipeline. Finally, a directory named Data containing the ECRDB61B-200 data set and the known E. coli binding sites should also be present to run the pipeline.

The required libraries to run the code can be seen in the list below:

- python~= 3.9.0
- ipython~= 8.6.0
- jupyter notebook~= 6.1.5
- matplotlib~= 3.6.0
- matplotlib-inline~= 0.1.6
- numpy~= 1.21.4
- pandas~= 1.3.5
- scipy~= 1.7.3
- seaborn~= 0.12.2
- tqdm~= 4.62.3