International Conference on Electrical, Computer and Communication Engineering (ECCE)
16–18 February 2017, Cox's Bazar, Bangladesh

RPPMD (Randomly Projected Possible Motif Discovery): An Efficient Bucketing Method for Finding DNA Planted Motif

Faisal Bin Ashraf*, Ali Imam Abir†, Md Sirajus Salekin‡ and M. A. Mottalib§
Department of Computer Science and Engineering, Islamic University of Technology, Dhaka, Bangladesh
*faisalashraf@iut-dhaka.edu, †aliimamabir@iut-dhaka.edu, ‡salekin@iut-dhaka.edu and §mottalib@iut-dhaka.edu

***Abstract*—DNA contains the information of structure and function of different molecules of any living being. Short repeating patterns in a DNA sequence, called Motifs, are useful to understand and analyze this information. Recent advancements in gene expression analysis already prompt the scientists to introduce a number of motif finding algorithms. Among different motif search problems, one of them is Planted Motif Search (PMS). In this paper, we have proposed an approximate algorithm for Planted Motif Search which at first generates all possible motif set and use a bucketing concept to find out the proper motifs from the whole dataset. Two benchmark datasets of DNA sequences are used to evaluate the performance of the proposed method and its comparative analysis with other approaches. Experimental results demonstrate that proposed bucketing approach is more robust than the other approximate algorithms providing more amounts of motifs within less amount of time. Most of the time, above 80% of the possible motifs of the given input set are found within a reasonable time.**

***Index Terms*—Approximate Algorithm, Bucketing, DNA Sequences, Motif Finding, Planted Motif Search**

I. Introduction

DNA (Deoxyribo Nucleic Acid) consists of bases of A (Adenine), T (Thymine), C (Cytosine), G (Guanine). A long sequence of A, T, C, G is called a DNA sequence. Every individual in every species has its own unique DNA sequence. In DNA sequence, there are short regions which are repeated. DNA can be mutated. In mutated DNA sequences, some of the bases are mutated. This short repeating patterns are called motif. Motif finding in biological sequences is very important as they help the biologists to understand better the structure and functions of the molecules represented by the sequences. If we give an overall outlook at the motif finding problem, it can be categorized into three types of motif finding problem which are Simple Motif Search (SMS), Edit distance based Motif Search (EMS) and Planted Motif Search (PMS). The goal of the Simple Motif Search (SMS) [1] is to find out all the motifs present in all sequences of the length 1 to a given length. Edit Distance based Motif Search [2] finds all those patterns which are present in a certain number of sequences. On the other hand, Planted Motif Search [3]–[5] aims to find the motifs which occur in every sequence. If we further explore the algorithm of Planted Motif Search (PMS), it can be observed that there are two types of approach - Exact and Approximate. In the Exact approach, all possible motifs which are present in all sequences are found, whereas Approximate approach cannot find all the possible motif but a good amount of motifs. Exact algorithm takes more time than Approximate approach as they tend to find all the possible motifs. On the contrary, Approximate approach is very time efficient. In this paper, we propose an approximate algorithm for PMS problem which finds out more motifs than other approximate algorithms. Most of the time, we get above 80% of the possible motifs of the given input set. Therefore, we hope our proposed algorithm will help biologists to find out the motifs in very short period of time.

Planted Motif Search (PMS) takes two inputs - the length of the desired motifs (l) and the number of mismatches considerable (d). That is why it is also called (l, d) - Planted Motif Search. All the sub-sequences of length l which occur in every sequence at least once are needed to be searched. If any sub-sequence of length l are present in any sequence which differs in maximum l places, that will be considered as an occurrence in that sequence. For example, if $ATTGCTGA$, $GCATTGAA$, $CATGCTTG$ are three sequences of nucleotides in three DNA sequences and we want to find out motifs of length 4 ($l = 4$) and maximum considerable mismatch is 1 ($d = 1$), then $ATTG$, $TTGC$ which are repeating pattern in these three sequences will be the planted motifs. Exact Motif Finding algorithms always find all the motifs of the given sequence set. But, PMS is found to be an NP-Hard problem. So, it takes exponential time to find out all the motifs. On the contrary, Approximate algorithms do not always find out all the motifs, but they are faster than exact algorithms.

Many exact algorithms [1], [3], [4], [6]–[12] have been proposed so far. In PMS1 [1], to find out motifs of length l with mismatches d at first, all possible $l-mers$ are generated from the input sequences and for each $l-mer$ all the possible neighbors having a hamming distance which is given is found out. Duplicate $l-mers$ are removed and finally the list is merged to find out all valid motifs. In PMS2 [1], to calculate (l, d) motifs at first, (k, d) motifs are found out using PMS1 [1] method where $k < l$. Then, $l-mer$ can be generated by merging $(l-k+1)$ number of those (k, d) motifs which occur starting from successive positions in each of the input strings. Later on, PMS3 [1] is proposed to find out the motifs of larger length, which used the idea of PMS1 [1]. All the motifs are merged one after another and generate the candidate motifs.

978-1-5090-5627-9/17/$31.00

Voting [8] is similar to PMS1. But it uses hashing where PMS1 [1] uses radix sort. RISSOTO [9] utilizes a suffix tree to find out (l, d) motifs. Later on, PMSPrune [10], an updated version of PMS, is introduced where there are two steps generating all possible candidate motif set and removing the non-motif candidates from the set. At first, a tree is built using these candidate motifs where every node is an $l-mer$ and the level of that particular node represents the number of mismatch between the original $l-mer$ of the first sequence and that candidate motif. Finally, the tree is pruned in the nodes where the number of mismatches is greater than d. PMS5 [11] is an improved algorithm of PMSPrune [10] which reduces the runtime of PMSPrune [10] by implementing Integer Linear Program. More modified algorithms of PMSPrune are also found such as qPMS7 [12] and PMS8 [4]. These algorithms always find all the planted motifs. But these algorithms cannot find motifs of all length.

Besides, a number of approximate algorithms [5], [13]–[17] are also proposed. In Pattern Branching [5] and Profile Branching [5], a local search technique is used where best neighbors of each $l-mer$ are kept. Consensus [13] is a scoring based greedy approach. MEME [14] uses a probabilistic method where expectation maximization is used. WINNOWER [15] is a graph theoretic approach where edges are established among the similar type of $l-mers$ with at most $2d$ mismatches where d is the allowed number of mismatches. SP-STAR [15] is a scoring method which uses a local improvement heuristic. Profiling based algorithm [16], at first, randomly chooses a sequence out of n sequences. Then the random position is selected in $n-1$ other sequences and generate $l-mer$ from those positions. Then profiling is done. Random Projection [17] algorithm projects all the $l-mers$ of the input sequences along k randomly chosen positions. After hashing all the $l-mers$ using the $k-mer$ of any $l-mer$, it groups all the $l-mers$ in the buckets using their hash value. Examine those buckets which contain a large number of $l-mers$. From those buckets, motifs are refined using expectation maximization. RPB-DC [18] also works like Random Projection [17]. But in this case, the positions for bucketing is not random. Rather, first three position are considered for bucketing and then characters are added one by one and checked out throughout the dataset. The process is continued until the expected length.

In this paper, we propose an approximate method which takes the idea of generating all candidate motif set of PMSPrune [10]. Then a bucketing approach is performed in both the candidate motif set and given dataset to find out the actual planted motifs.

The rest of the paper is organized as follows. Section -II discusses our proposed method. Section-III contains experimental results and discussion and Section IV concludes the summary of this paper.

II. Proposed Method

A. Notations and Definitions

In this section, some notations and definitions are introduced which will help to describe our algorithm clearly.

Definition 1. *Given a set of sequences* $S = S_1, S_2, S_3, .., S_n$ *over an alphabet set A,T,C,G such that* $|S_i| = m$ *and two integer value* l *and* d, $0 \leq d < l < m$, *we define Planted Motif Search(PMS) as to find out a subsequence* x *such that* $|x| = l$ *and every sequence in* S *has a subsequence* x_i *such that* $|x_i| = l$ *and x differs from* x_i *in at most* d *places.*

Definition 2. *A string x having length of l is called an* $l-mer$.

Definition 3. $D_H(x, y)$ *is the number of mismatches between strings x and y over alphabet set A,T,C,G.*
$D_H(x, y)$ *:= number of mismatches between x and y*

Definition 4. $B_d(x)$ *is the set of all neighbors of string x over alphabet set A,T,C,G.*
$B_d(x)$ *:=* $\{z \mid D_H(x, z) \leq d\}$

Definition 5. C_{bucket} *is the list of those buckets which are obtained after projecting all* $l-mers$ *of candidate motif set along* k *randomly chosen positions.*
C_{bucket}*:=* $\{ C_{AAAA}, C_{AAAT}, \ldots, C_{TTTT} \}$
This given C_{bucket} *is the list of buckets after projecting along 4 randomly chosen position.*

Definition 6. S_{bucket} *is the list of those buckets which are obtained after projecting all* $l-mers$ *of the given input sequences set along* k *randomly chosen positions.*
S_{bucket}*:=* $\{ S_{AAAA}, S_{AAAT}, \ldots, S_{TTTT} \}$
This given S_{bucket} *is the list of buckets after projecting along 4 randomly chosen position.*

Definition 7. S_x *is correspondent bucket of* C_y *, where x = y. For example,* S_{ATTG} *is the correspondence bucket of* C_{ATTG} .

Definition 8. L_x *is the list of all* $l-mers$ *in* C_x .

The idea of our proposed method is based on following observations.

Observation 1: $B_{d(x)}$ for all $l-mer$ x in S_1 of the given sequence set $S = \{S_1, S_2, \ldots, S_n\}$ contains all possible motifs for the set S as well as some more $l-mers$ which are not motif for S.

Observation 2: Bucketing all the $l-mers$ of the given set S along k randomly chosen position, for $k < l-d$, will cluster all those $l-mers$ which have no mismatch in those k positions in the same bucket.

From observation 1, we realize that we can build a candidate motif set for a given input set S which contains all $l-mer$ neighborhoods of first sequence S_1. Observation 2 states that, using projection along randomly chosen k position will give us the clusters of the input sets $l-mers$.

B. Our Proposed Method

From the observations, we have proposed a novel method for finding DNA planted motif which first generates all candidate motif set and then extracts original motifs. Steps of our proposed method are shown in Figure 1 using a flowchart.

$Step-1$: Generate all $l-mers$ from first input sequence. The planted motifs are present in this $l-mers$ set. Because any planted motif must be present in first sequence with at most d mismatches. Generated $l-mers$ set from first sequence

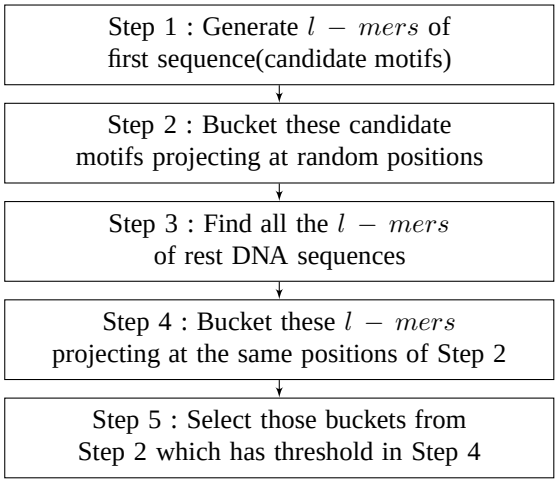

Fig. 1. Steps of the proposed method

ensures that the output motifs are present in the set and termed as candidate motif's set.

$Step-2$: Cluster the $l-mers$ from Step-1 in buckets using random projection along k randomly chosen position where $k < l - d$. Then we will get some buckets of all $l - mers$ of candidate motif set, i.e, C_{bucket}.

$Step - 3$: Generate a set of all $l - mers$ of all given sequences except the first sequence. These $l - mers$ are the subsequences of the input sequences of length l and their neighbors having maximum d mismatches.

$Step-4$: Project all the $l-mers$ from Step-3 along same k positions which positions were selected for bucketing in Step-2. Thus we will get some buckets of $l - mers$ of input sequences, i.e., S_{bucket} .

$Step - 5$: Output all the $l - mers$ contained in bucket C_{bucket} where its correspondent S_{bucket} has size greater than threshold. The $l - mers$ in those C_{bucket} are the output planted motifs of length l having at most d mismatches of the given input sequences.

$Motif := \cup_{\forall x} L_x$ where size of $S_x \geq S_{threshold}$

We need to define the threshold value for the buckets of the input sets. We set this value as the average bucket size, i.e.

$S_{threshold}$ = $\frac{n(m-l+1)}{4^k}$
where, n = Total number of sequences in S
m = length of each sequence (S_i)
k = randomly chosen positions

The pseudo code of our proposed algorithm is shown in Algorithm 1.

In the proposed approach, buckets from dataset ensure that there is no mismatch in the bucketing positions for all the $l - mers$ in a bucket. Besides, the selected buckets have the threshold number of $l-mers$ which will assure that the order of A, T, C, G is present in all the given sequences. As a result, all the possible candidate motifs are provided which certainly have all the original motifs for the dataset. From these, we can exclude the buckets of candidate motifs and refine motifs from the other buckets by checking the threshold value. Moreover, proposed method takes less time because it does not need to further calculate whether the $l - mers$ are original motifs or not. Because they are already candidate motifs following the definition of motif and we need only to exclude the non-motif $l - mers$.

Algorithm 1 Proposed algorithm for planted motif search

```
Q ← φ
for each l − mer x of S_1
    Compute B_d(x)
    Q ← Q ∪ B_d(x)
end for
k = l − d − 1
Motif ← φ
for 1 to m
    generate k random number R_1, R_2, ..., R_k where R_1 ≠ R_2 ≠ ... ≠ R_k
    for each l − mer y in Q
        String sbucket = y[R_1] + y[R_2]+ ... + y[R_k]
        add y into bucket S_sbucket
    end for
    P ← φ
    for 1 to 4^k
        if size of S_z ≥ S_threshold then
            P ← P ∪ l − mers of C_z
        end if
    end for
    if Motif ← φ then
        Motif ← P
    else
        Motif ← Motif ∩ P
    end if
end for
```

III. Results and Discussion

To evaluate the performance of the proposed method, two benchmark datasets [19], [20] are used. Robustness of the proposed method is evaluated in case of motif length means different instances and accuracy.

A. Dataset

The performance of the proposed method for different instances is checked using dataset [19]. There are 52 datasets - 6 from fly, 26 from human, 12 from mouse and 8 from yeast. Number of sequences in the datasets vary from 5 to 35 and length of each sequence vary from 1000 to 2500. All the sequences in the dataset are in FASTA format. Besides, for comparing the accuracy of the proposed method with other approximate algorithms the real dataset [20] are used which consists of the DNA sequences of yeast and fly.

B. Experimental Discussion

The experimental analysis of the proposed method is performed in two phases. Firstly, the performance is evaluated in case of different motif length. In Table I, different datasets from the benchmark dataset [19] and different value of (l, d) are used to analysis the robustness of the proposed method. From this table I, it can be observed that our method works very close to exact approaches when the dataset length is high and the number of sequences is large. For (6, 2) motif length it shows its best results for each case. Even for higher length like (7, 2) or (8, 3) still, it shows comparatively better results. Besides, the proposed approach takes a reasonable amount of time to find the different length motif. Secondly, our proposed method is also compared with different approximate approaches in case of the number of motifs found. The experiments are performed using the benchmark dataset [20]. From Table II, it is observable that our method can find about 80% of the motifs of the dataset whereas the closest success rates are 52% and 57% in case of MEME:ZOOP-DC [21] and RPB-DC [18] respectively.

TABLE I
EXPERIMENTAL RESULTS ON DATA SET [19]

Dataset	Number of Sequences	Length of Each Sequences	(l,d)	Time (s)	Found Motif (%)
dm01r	5	1500	(6,2)	1.81	87
			(7,2)	5.804	55
			(8,3)	39.55	58
dm05r	3	2500	(6,2)	2.574	99
			(7,2)	7.269	97
			(8,3)	51.537	94
hm01r	18	2000	(6,2)	2.73	93
			(7,2)	7.708	88
			(8,3)	50.135	85
hm03r	10	1500	(6,2)	1.998	74
			(7,2)	6.302	63
			(8,3)	41.565	60
hm20r	35	2000	(6,2)	3.323	98.8
			(7,2)	9.84	98.14
			(8,3)	52.089	97.3
mus04r	6	1000	(6,2)	1.4	76.4
			(7,2)	5.08	60.3
			(8,3)	35.45	60
mus10r	13	1000	(6,2)	1.42	78
			(7,2)	5.383	68
			(8,3)	35.864	60

TABLE II
COMPARISON OF NUMBER OF DISCOVERED MOTIFS OF OUR PROPOSED METHOD AND OTHER APPROXIMATE ALGORITHMS ON DATASET [20]

Algorithms	Found Motif ((%)
PhyloCon [22]	12
PhyME [20]	13
MEME:ZOOPS [14]	25
PhyloGibbs [23]	35
Kellis et al. [24]	36
Converge [25]	44
MEME:OOPS- DC [21]	47
PRIORITY- DC [26]	49
MEME:ZOOP- DC [21]	52
RPB-DC [18]	57
Proposed Method	**80**

IV. CONCLUSION

In this paper, an approximate method for Planted Motif Search (PMS) problem is proposed which is efficient in consideration of run time as well as finding more number of motifs compared to other approximate algorithms. Experimental results exhibit that the proposed approach finds around 80% of the motifs within less amount of time. Although, some of the motifs are inappropriately clustered into wrong buckets due to the exclusion of some candidate motifs by the threshold value, but still a better motif finding rate is achieved. As a result, we hope this algorithm will ensure a better motif finding approach for a particular DNA sequence to help biologists.

REFERENCES

[1] S. Rajasekaran, S. Balla, C.-H. Huang, V. Thapar, M. R. Gryk, M. W. Maciejewski, and M. R. Schiller, "Exact algorithms for motif search." in *APBC*, 2005, pp. 239–248.

[2] S. Pal and S. Rajasekaran, "Improved algorithms for finding edit distance based motifs," in *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 537–542.

[3] H. M. Martinez, "An efficient method for finding repeats in molecular sequences," *Nucleic acids research*, vol. 11, no. 13, pp. 4629–4634, 1983.

[4] M. Nicolae and S. Rajasekaran, "Efficient sequential and parallel algorithms for planted motif search," *BMC bioinformatics*, vol. 15, no. 1, p. 1, 2014.

[5] A. Price, S. Ramabhadran, and P. A. Pevzner, "Finding subtle motifs by branching from sample strings," *Bioinformatics*, vol. 19, no. suppl 2, pp. ii149–ii155, 2003.

[6] A. Brāzma, I. Jonassen, J. Vilo, and E. Ukkonen, "Predicting gene regulatory elements in silico on a genomic scale," *Genome research*, vol. 8, no. 11, pp. 1202–1215, 1998.

[7] S. Sinha and M. Tompa, "A statistical method for finding transcription factor binding sites." in *ISMB*, vol. 8, 2000, pp. 344–354.

[8] F. Y. Chin and H. C. Leung, "Voting algorithms for discovering long motifs." in *APBC*, 2005, pp. 261–271.

[9] N. Pisanti, A. M. Carvalho, L. Marsan, and M.-F. Sagot, "Risotto: Fast extraction of motifs with mismatches," in *Latin American Symposium on Theoretical Informatics*. Springer, 2006, pp. 757–768.

[10] J. Davila, S. Balla, and S. Rajasekaran, "Fast and practical algorithms for planted (l, d) motif search," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 4, pp. 544–552, 2007.

[11] H. Dinh, S. Rajasekaran, and V. K. Kundeti, "Pms5: an efficient exact algorithm for the (l, d)-motif finding problem," *BMC bioinformatics*, vol. 12, no. 1, p. 410, 2011.

[12] H. Dinh, S. Rajasekaran, and J. Davila, "Qpms7: A fast algorithm for finding (l, d)-motifs in dna and protein sequences," *PloS one*, vol. 7, no. 7, p. e41425, 2012.

[13] G. Z. Hertz and G. D. Stormo, "Identifying dna and protein patterns with statistically significant alignments of multiple sequences." *Bioinformatics*, vol. 15, no. 7, pp. 563–577, 1999.

[14] T. L. Bailey and C. Elkan, "The value of prior knowledge in discovering motifs with meme." in *ISMB*, vol. 3, 1995, pp. 21–29.

[15] P. A. Pevzner, S.-H. Sze *et al.*, "Combinatorial approaches to finding subtle signals in dna sequences." in *ISMB*, vol. 8, 2000, pp. 269–278.
[16] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, J. C. Wootton *et al.*, "Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment," *SCIENCE-NEW YORK THEN WASHINGTON-*, vol. 262, pp. 208–208, 1993.
[17] J. Buhler and M. Tompa, "Finding motifs using random projections," *Journal of computational biology*, vol. 9, no. 2, pp. 225–242, 2002.
[18] M. Galib, N. Hasan, M. A. Rahman, and M. A. Mottalib, "Rpb-dc: Motif discovery using randomly projected bucketing (rpb) and discriminative conservation (dc) based prior," *icita.org*, vol. 15, 2015.
[19] M. Tompa, N. Li, T. L. Bailey, G. M. Church, B. De Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent *et al.*, "Assessing computational tools for the discovery of transcription factor binding sites," *Nature biotechnology*, vol. 23, no. 1, pp. 137–144, 2005.
[20] S. Sinha, M. Blanchette, and M. Tompa, "Phyme: a probabilistic algorithm for finding motifs in sets of orthologous sequences," *BMC bioinformatics*, vol. 5, no. 1, p. 1, 2004.
[21] T. L. Bailey, M. Bodén, T. Whitington, and P. Machanick, "The value of position-specific priors in motif discovery using meme," *BMC bioinformatics*, vol. 11, no. 1, p. 1, 2010.
[22] T. Wang and G. D. Stormo, "Combining phylogenetic data with co-regulated genes to identify regulatory motifs," *Bioinformatics*, vol. 19, no. 18, pp. 2369–2380, 2003.
[23] R. Siddharthan, E. D. Siggia, and E. Van Nimwegen, "Phylogibbs: a gibbs sampling motif finder that incorporates phylogeny," *PLoS Comput Biol*, vol. 1, no. 7, p. e67, 2005.
[24] M. Kellis, N. Patterson, M. Endrizzi, B. Birren, and E. S. Lander, "Sequencing and comparison of yeast species to identify genes and regulatory elements," *Nature*, vol. 423, no. 6937, pp. 241–254, 2003.
[25] K. D. MacIsaac, T. Wang, D. B. Gordon, D. K. Gifford, G. D. Stormo, and E. Fraenkel, "An improved map of conserved regulatory sites for saccharomyces cerevisiae," *BMC bioinformatics*, vol. 7, no. 1, p. 1, 2006.
[26] R. Gordan, L. Narlikar, and A. J. Hartemink, "Finding regulatory dna motifs using alignment-free evolutionary conservation information," *Nucleic Acids Research*, vol. 38, no. 6, pp. e90–e90, 2010.