Efficient Algorithms for Finding Edit-Distance Based Motifs

Peng Xiao, Xingyu Cai, and Sanguthevar Rajasekaran(✉)

Department of Computer Science and Engineering,
University of Connecticut, Storrs, CT, USA
{peng.xiao,xingyu.cai,sanguthevar.rajasekaran}@uconn.edu

Abstract. Motif mining is a classical data mining problem which aims to extract relevant information and discover knowledge from voluminous datasets in a variety of domains. Specifically, for the temporal data containing real numbers, it is formulated as time series motif mining (TSMM) problem. If the input is alphabetical and edit-distance is considered, this is called Edit-distance Motif Search (EMS). In EMS, the problem of interest is to find a pattern of length l which occurs with an edit-distance of at most d in each of the input sequences.

There exists some algorithms proposed in the literature to solve EMS problem. However, in terms of challenging instances and large datasets, they are still not efficient. In this paper, EMS3, a motif mining algorithm, that advances the state-of-the-art EMS solvers by exploiting the idea of projection is proposed. Solid theoretical analyses and extensive experiments on commonly used benchmark datasets show that EMS3 is efficient and outperforms the existing state-of-the-art algorithm (EMS2).

Keywords: Sequence analysis · Edit-distance motif · Projection

1 Introduction

Effective data mining algorithms when applied on biological data can reveal crucial information that could lead to accurate diagnosis, drug development, and disease treatment. One set of such mining algorithms are referred to as motif mining (or motif search) algorithm. These algorithms look for information that is closely preserved across species. For example, a piece of gene segment may appear exactly or with minor differences across different species. Extracting such information is very meaningful in numerous applications, such as the determination of open reading frames, identification of gene promoter elements, location of RNA degradation signals, and the identification of alternative splicing sites. Many motif mining models have been proposed.

This paper focuses on the Edit-distance based Motif Search (EMS) model. EMS is defined as follows: Inputs are two integers l and d, and n biological strings over the alphabet Σ of a finite size. Each string is of length m. The problem is to find all the strings of length l that appear in each of the n input strings with

I. Holmes et al. (Eds.): AlCoB 2019, LNBI 11488, pp. 212–223, 2019.
https://doi.org/10.1007/978-3-030-18174-1_16

the Levenshtein distance (or edit-distance) of at most d. Biologists may also be interested in motifs that occur in a fraction of the input strings. The problem of identifying such motifs is known as quorum Edit-distance Motif Search (qEMS). In this case, an extra input parameter q is provided. The problem is to identify all the (l, d, q)-motifs, that is, all (l, d)-motifs that occur in at least qn of the input strings. The standard EMS problem becomes a special case of qEMS when $q = 1$.

In EMS, the edit-distance is used to bound the variability of the pattern across the biological sequences. It can include substitution, insertion and deletion. If only substitution is allowed (*i.e.*, the aim is to find the strings of length l with a Hamming distance of at most d in each of the input sequences), this simplified version of the problem is named as Planted Motif Search (PMS). There are many studies on PMS problems (see *e.g.*, [7,8,16]). EMS is more challenging than PMS because EMS is more general.

It is known that there is a polynomial time reduction from the Closest Substring problem to PMS [8]. Since the Closest Substring problem is NP-Hard [3,4,6], PMS problem is also NP-Hard. EMS is also NP-hard since PMS is a special case of EMS. Therefore, it is of pressing need to develop efficient exact algorithms for EMS problems.

EMS and its variations have been studied since a long time ago. Back in 1998, the authors in [13] proposed an algorithm to find approximate repeats from a long DNA sequence, allowing general insertions and deletions. This is an approximate algorithm. Suffix tree based algorithms are also developed to find approximate repeated or common motifs [14]. The algorithms proposed in [14] can be extended to deal with gaps but the authors did not implement it for edit-distance but only for Hamming distance. The authors in [1] proposed a new algorithm to extract common motifs using the techniques for extracting approximate non-tandem repeats and they also implemented Sagot's algorithm in [14] and did a comparison. The result shows that their algorithm has less false positive motifs and is also more efficient for finding moderately long motifs.

Algorithms proposed in the literature above are part of the early stage studies of the EMS problem. However, the first formal definition of EMS is given in [11] although the authors did not explicitly name it as EMS. They also proposed a deterministic (DMS) algorithm that runs in time $O(n^2 m P^D |\Sigma|^D)$ using $O(nmD + P^D|\Sigma|^D)$ space (P and D are motif length and maximum allowed edit-distance, respectively). A Monte Carlo algorithm with a run time of $O(((n^2 m^2 \log n)/q)D + gmnD)$ is also proposed where g is the number of P-mers that occur in q or more sequences in the database. Following this definition, Pathak *et al.* [10] proposed EMS1 which naturally extends the data structure of d-neighborhood tree from the PMS problem and they evaluated this algorithm on synthetic datasets as well as real datasets. However, one drawback of EMS1 is that it generates too many repeated neighborhoods which takes up a huge memory and also the (l, d) instances it can solve are very limited. To alleviate this problem, Pal *et al.* [9] proposed EMS2. They used wildcard characters to compactly represent the neighborhood tree and also proposed 9 rules to avoid

duplications of candidate motifs. EMS2P, which is a parallel algorithm, was also developed and tested on a multi-core machine. Experimental results showed that EMS2 is faster than EMS1 and the parallel version has a good scaling performance.

In this paper, an improved algorithm, EMS3, is proposed to further advance the state-of-the-art EMS solvers. Theoretical study and extensive experimental tests are performed. The rest of the paper is organized as follows. Section 2 presents the proposed algorithm. In Sects. 3 and 4, theoretical analyses and empirical studies of EMS3 are provided. Finally, a brief summary concludes the paper in Sect. 5.

2 EMS3: An Improved Algorithm

2.1 Overview of the Algorithm

EMS3 has 5 steps as follows.

Step 1: Divide Choose an appropriate value of $\epsilon_1 \in (0, 1]$. Let $n' = n * \epsilon_1$. Randomly select n' sequences from the input I. Let this set be I_1. Let $I_2 = I - I_1$.

Step 2: Compress Choose an appropriate value of $\epsilon_2 \in (0, 1]$. Let $|\Sigma'| = |\Sigma| * \epsilon_2$. Compress I_2 by projecting Σ to Σ'. Specifically, every $1/\epsilon_2$ characters in Σ will be projected to a single character in Σ'. For example, if $\Sigma = \{A, C, G, T\}$ and $\epsilon_2 = 1/2$, A, C will be projected to A while G, T will be projected to C. $\Sigma' = \{A, C\}$. This process is also called as encoding.

Step 3: Solve the subproblems Run existing EMS solver on I_1 and I_2. Let the outputs be C_1 and C_2, respectively. Note that the strings from C_2 are in the domain of Σ'. Both C_1 and C_2 are sorted.

Step 4: Merge One idea is to expand the strings in C_2. Let the resultant string set be C_2'. C_2' is in the domain of Σ. The intersection of C_1 and C_2', denoted as C, will be the final candidate set. However, this solution, to a great extent, wipes out the advantage of reducing the alphabet size in the second step because the size of C_2' will be too large. Moreover, C_2' is not sorted any more. Another round of radix sort needs to be performed on C_2' to merge these two sets. A better idea to salvage the execution time is to encode the strings in C_1 in the same way as discussed above. For an l-mer $u \in C_1$, the encoded string is u'. Check if it is in C_2. If it is, add u to C. Note that since C_2 is sorted, a binary search can be performed.

Step 5: Verify Let the output of EMS3 be O. For every l-mer $v \in C$, check if it is an (l, d) edit-distance motif in the remaining sequences, *i.e.*, I_2. If so, add v to O. Three algorithms are proposed for this step.

VerifyMotif_1, *VerifyMotif_2* and *VerifyMotif_3* are the pseudocodes. For all these three algorithms, the inputs are two integers l and d, a set of sequences $\{S_i\}$ $(i = 1, 2, \ldots, n)$, and an l-mer v. The output is a boolean flag indicating whether v is the target (l, d)-motif.

Algorithm 1. VerifyMotif_1(l, d, v, $\{S_i\}$)

$i \leftarrow 1$; $isMotifSeq \leftarrow$ False; $isMotifSeqs \leftarrow$ True;
while $i \leqslant |\{S_i\}|$ **do**
  for $k \leftarrow l - d$ *to* $l + d$ **do**
    $subS_i \leftarrow$ the collection of all substrings of length k in S_i;
  for *every* $x \in subS_i$ **do**
    $e(v, x) \leftarrow$ EditDistance(v, x); `// Dynamic programming to compute edit-distance between` v `and` x
    if $e(v, x) \leqslant d$ **then**
      $isMotifSeq \leftarrow$ True;
      $i \leftarrow i + 1$; Break;
  if $isMotifSeq =$ *False* **then**
    $isMotifSeqs \leftarrow$ False;
    Break;
return $isMotifSeqs$;

Algorithm 2. VerifyMotif_2(l, d, v, $\{S_i\}$)

$i \leftarrow 1$; $isMotifSeq \leftarrow$ False; $isMotifSeqs \leftarrow$ True;
$T \leftarrow v$'s d-neighborhood; `// Call GenerateNeighborhoodTree in [9]`
while $i \leqslant |\{S_i\}|$ **do**
  for *every* $w \in T$ **do**
    $isMotifSeq \leftarrow$ ExactPatternMatch(w, S_i); `// KMP algorithm [5] to check if` w `appears exactly in` S_i
    if $isMotifSeq =$ *True* **then**
      $i \leftarrow i + 1$; Break;
  if $isMotifSeq =$ *False* **then**
    $isMotifSeqs \leftarrow$ False;
    Break;
return $isMotifSeqs$;

Algorithm 3. VerifyMotif_3(l, d, v, $\{S_i\}$)

$i \leftarrow 1$; $isMotifSeq \leftarrow$ False; $isMotifSeqs \leftarrow$ True;
$T \leftarrow v$'s d-neighborhood; `// Call GenerateNeighborhoodTree in [9]`
while $i \leqslant |\{S_i\}|$ **do**
  for $k \leftarrow l - d$ *to* $l + d$ **do**
    $subS_i \leftarrow$ a collection of substrings of length k in S_i;
  for *every* $x \in subS_i$ **do**
    if $x \in T$ **then** `// Perform binary search`
      $isMotifSeq \leftarrow$ True;
      $i \leftarrow i + 1$; Break;
  if $isMotifSeq =$ *False* **then**
    $isMotifSeqs \leftarrow$ False;
    Break;
return $isMotifSeqs$;

2.2 Why Project the Alphabet

It is desirable to reduce the size of the input while maintaining the accuracy of the algorithm. One way is to project the high dimensional space of the input data into a low dimensional one. The authors in [2] use this technique to find the planted motifs in PMS problems. They randomly choose k selected positions of each l-mer x as a hash function $h(x)$. In other words, they project the motif length from l to k. In [12,16,17], the authors use the idea of random sampling. They randomly select n' out of n sequences and run PMS solvers on the sample dataset. This can be considered as a projection of the number of biological sequences from n to n'.

To the best of the authors' knowledge, the idea of alphabet projection has not been employed before to solve motif mining problems. It is believed that the alphabet size has a great impact on the time complexity of EMS algorithms. For example, in [11], the authors proposed an algorithm to solve the qEMS problem that runs in $O(n^2ml^d|\Sigma|^d)$. In [10], the authors proposed EMS1 which has a time complexity of $O(mn(4l|\Sigma|)^d + |\Sigma|^l)$. Compared to sampling n, projecting the alphabet to a smaller size will greatly reduce the running time.

2.3 Correctness of the Algorithm

It is easy to see that EMS3 is a deterministic algorithm that always output the correct motifs. An important question is whether EMS3 misses any true motifs. The answer is no. Assume that the set of true motifs is G. It can be proved that after the merge step, the candidate motif set C is a superset of the true motif set G, *i.e.*, $G \subseteq C$.

Please note due to page limit, from this point, proof of the lemmas and theorems are omitted. Interested readers can ask the authors for details.

Lemma 1. *Let l_1 and l_2 be strings on Σ and let the edit-distance between l_1 and l_2 be d. Let l_1' and l_2' be compressed strings of l_1 and l_2 using the projection technique discussed above from Σ to Σ' ($|\Sigma'| \leqslant |\Sigma|$). The edit-distance between l_1' and l_2', denoted as d', is no more than d.*

Theorem 2. $G \subseteq C$.

3 Analysis of EMS3

3.1 Time Complexity Analysis

Expected Number of Candidate Motifs. The expected number E of candidate motifs is a function of $m, n, l, d, |\Sigma|$, and is derived in [9], to which the interested reader is referred for details. In that paper, the Eqs. 1, 2 and 3 below, with δ, β, α, and q acting as dummy variables, lead to an expression for E in Eq. 4.

$$N(\delta, \beta, \alpha, l, |\Sigma|, q) = \binom{l+q}{\delta}\binom{l+q-\delta}{\beta}\binom{l+q-\delta+\alpha}{\alpha}|\Sigma|^{\alpha}(|\Sigma|-1)^{\beta}. \quad (1)$$

$$P(l,|\Sigma|,d,q) = \sum_{\delta=\max\{0,q\}}^{\frac{d+q}{2}} \frac{N(\delta,d+q-2\delta,\delta-q,l,|\Sigma|,q)}{|\Sigma|^{l+q}}. \tag{2}$$

$$R(m,l,d,|\Sigma|) = \Pi_{q=-d}^{d}(1-P)^{m-l-q+1}. \tag{3}$$

$$E(m,n,l,d,|\Sigma|) = |\Sigma|^{l}(1-R)^{n}. \tag{4}$$

The expected sizes of C_1 and C_2 can be written as:

$$E(|C_1|) = E(m,n',l,d,|\Sigma|). \tag{5}$$

$$E(|C_2|) = E(m,(n-n'),l,d,|\Sigma'|). \tag{6}$$

An l-mer has a probability of $p_1 = (1-R(m,l,d,|\Sigma|))^{n'}$ to be in C_1. If it is encoded, it has a probability of $p_2 = (1-R(m,l,d,|\Sigma'|))^{n-n'}$ to be in C_2. Every $(1/\epsilon_2)^l$ l-mers in Σ will be projected to a single l-mer in Σ'. Therefore, the expected number of l-mers in C is:

$$E(|C|) = |\Sigma|^{l}p_1p_2/\epsilon_2^{l}. \tag{7}$$

In order to make sure that the expected number of candidate motifs is reduced, it is desirable to have:

$$p_2/\epsilon_2^{l} < 1. \tag{8}$$

However, it does not mean that p_2/ϵ_2^{l} should be as small as possible. When ϵ_2 is large, the size of the candidate motif set is small thus reducing the running time in Step 5 of EMS3. However, the running time of Step 3 increases because of a relatively large alphabet size.

Time Complexity of EMS2. Time complexity of EMS2 is not given in its original paper [9]. It can be shown that the overall time complexity of EMS2 is:

$$T_{EMS2} = O(mndl^{d+1}|\Sigma|^{d}). \tag{9}$$

Note that this may be larger than the time complexity in [10,11]. This is because unlike [10,11], in EMS2 and EMS3, assumption that the motifs of interest should come exactly from one of the input sequences is removed to make this problem more general. However, this only represents the worst time complexity. A lot of branches of the neighborhood tree can be pruned because of the rules proposed in [9]. Therefore, the actual running time is much less. But it is hard to estimate how many branches will be pruned.

Time Complexity of Verifying Candidate Motifs. There are three algorithms to verify the candidate motifs. The first algorithm uses dynamic programming to compute the edit-distance between an l'-mer ($l - d \leqslant l' \leqslant l + d$) and an l-mer. Therefore the time taken is:

$$T_{verify\_1} = |C|n \sum_{l'=l-d}^{l+d} O(l'l(m - l' + 1)) = O(|C|mndl^2).$$

$|C|$ is the size of the candidate motif set. An expected number of the candidate motifs $E(|C|)$ can be found in Eq. 7.

The second algorithm will generate the d-neighborhood tree and use a linear time complexity algorithm to locate the neighborhoods within the input sequence. It is known that the number of d-neighborhoods of an l-mer is $O(l^d|\Sigma|^d)$. Thus, time taken in this algorithm is (assuming $d < l < m$):

$$T_{verify\_2} = |C|nO(l^d|\Sigma|^d(m + l)) = O(|C|mnl^d|\Sigma|^d).$$

The third algorithm also generates d-neighborhood tree but tries to locate the k-mer ($l - d \leqslant k \leqslant l + d$) from the input sequences in the tree. Time complexity for this algorithm is:

$$T_{verify\_3} = |C|nO(\sum_{k=l-d}^{l+d} (m - k + 1)\log(l^d|\Sigma|^d)) = O(|C|mnd^2(\log l + \log|\Sigma|)).$$

These are only upper bounds of the time taken in each algorithm. It looks like that the second algorithm takes the longest time. However, in generating the neighborhood tree, a lot of branches are pruned. The neighborhoods are also sorted and duplications are removed. Therefore the actual number of neighborhoods is far less than $l^d|\Sigma|^d$.

Time Complexity of EMS3. Step 1 and 2 take time that is negligible. Time complexities of Step 3 and 5 are already analyzed. Step 4 takes time $O(l|C_1|\log|C_2|)$. An expected number of $E(|C_1|)$ and $E(|C_2|)$ can be found in Eqs. 5 and 6. Therefore, assuming using *VerifyMotif_1* in the final step to verify the candidate motifs, the time complexity of EMS3 is:

$$T_{EMS3} = O(mdl^{d+1}(n'|\Sigma|^d + (n - n')|\Sigma'|^d) + l|C_1|\log|C_2| + |C|mndl^2). \quad (10)$$

where $n' = n * \epsilon_1, |\Sigma'| = |\Sigma| * \epsilon_2$.

3.2 How to Choose ϵ_1 and ϵ_2

Equation 8 shows the first guideline to choose ϵ_1 and ϵ_2. Generally, in a divide and conquer algorithm, it is desirable to split the input into nearly equal halves so that the performance of the algorithm is the best. However, in EMS3, in order

to balance the run times of the two subproblems, their running times should be in the same scale. Therefore, ϵ_1 is smaller than 1/2. Solve $n'|\Sigma|^d = \Theta((n-n')|\Sigma'|^d)$:

$$\epsilon_1 = \Theta((1-\epsilon_1)\epsilon_2^d). \tag{11}$$

It is also noteworthy to point out that in conventional notion of divide and conquer, the same algorithm is applied recursively to the subproblems whereas in EMS3, the subproblems are solved non-recursively using EMS2 solver.

3.3 A Discussion on qEMS and Approximate EMS3

EMS3 can be extended to qEMS problems as well. Under such circumstances, besides ϵ_1 and ϵ_2, another parameter ϵ_3 $(0 < \epsilon_3 \leqslant 1)$ is needed.

Theorem 3. *If an l-mer x is an (l, d, q) edit-distance motif on Σ, then the encoded l-mer x' is also an (l, d, q) edit-distance motif on Σ' using the projection technique as discussed, with $|\Sigma'| \leqslant |\Sigma|$.*

Theorem 4. *If an l-mer x is an (l, d, q) edit-distance motif on n sequences, then it is also an $(l, d, q\epsilon_3)$ edit-distance motif on $n\epsilon_1$ $(0 < \epsilon_1 \leqslant 1)$ sequences as long as the following condition is satisfied: $\epsilon_1(1-\epsilon_3 q) \geqslant 1-q$.*

Theorem 5. *If an l-mer x is an (l, d, q) edit-distance motif on n sequences, then it is also an $(l, d, q\epsilon_3)$ edit-distance motif on $n\epsilon_1$ $(0 < \epsilon_1 \leqslant 1)$ sequences with a high probability as long as: $(1-\epsilon_3)^2\epsilon_1 \geqslant \frac{2\beta \ln(n)}{nq}$. β is a constant. A high probability means that the probability is higher than $1 - n^{-\beta}$.*

When n is small, ϵ_1 and ϵ_3 can be chosen according to Theorem 4 to make sure the algorithm can always output the correct answer. When n is large, ϵ_1 and ϵ_3 can be chosen according to Theorem 5. In this case, EMS3 becomes a randomized algorithm.

If Step 5 of EMS3 is removed, EMS3 becomes an approximate algorithm. It is known that the candidate motifs set C is a superset of the true motifs set G.

Theorem 6. *If Step 5 of EMS3 is removed, then it becomes an approximate algorithm with an expected approximation ratio of $E(|C|)/E(m, n, l, d, |\Sigma|)$. $E(|C|)$ and $E(m, n, l, d, |\Sigma|)$ can be found in Eqs. 7 and 4, respectively.*

4 Experimental Evaluations of EMS3

Extensive experiments on existing standard benchmark datasets are performed to evaluate EMS3. All the algorithms are evaluated on a Dell Precisions Workstation T7910 running RHEL 7.0 on two sockets each containing 8 Dual Intel Xeon Processors E5-2667 (8C HT, 20 MB Cache, 3.2 GHz) and 256 GB RAM. *VerifyMotif_1* is used as the algorithm to verify the candidate motifs.

Table 1. Size of candidate motif sets with different n' and $|\Sigma|'$

(a) $(l, d) = (8, 1)$

n'	$\lvert C_1\rvert$	$\lvert C\rvert$	
		$\lvert\Sigma\rvert' = 2$	$\lvert\Sigma\rvert' = 3$
2	6762	6762	4549
4	667	667	503
6	74	74	58
8	7	7	5
10	2	2	1

(b) $(l, d) = (12, 2)$

n'	$\lvert C_1\rvert$	$\lvert C\rvert$	
		$\lvert\Sigma\rvert' = 2$	$\lvert\Sigma\rvert' = 3$
2	72763	72749	21157
4	389	389	99
6	9	9	5
8	1	1	1
10	1	1	1

(c) $(l, d) = (16, 3)$

n'	$\lvert C_1\rvert$	$\lvert C\rvert$	
		$\lvert\Sigma\rvert' = 2$	$\lvert\Sigma\rvert' = 3$
2	721497	719577	18794
4	203	203	7
6	1	1	1
8	1	1	1
10	1	1	1

4.1 Synthetic Datasets

Following the tradition, n $(= 20)$ DNA sequences of length m $(= 600)$ each are generated, where each character is independent and identically distributed (i.i.d.) over the alphabet under concern ($\Sigma = \{A, C, G, T\}$). A random string M of length l is randomly generated as the target motif. Besides, a d-neighborhood string is planted in each of the n sequences. In addition to the motif planted, there could be other motifs that occur by random chance. Challenging instances of $(l, d) = (8, 1), (12, 2), (16, 3)$ are tested. $(l, d) = (20, 4)$ is not tested because EMS2 cannot complete it within stipulated 72 h.

n' varies from 2 to 10. $|\Sigma|' = 2$ and 3. From Table 1, it can be seen that when $|\Sigma|' = 2$, the size of candidate motif set $|C|$ is not reduced. In fact, it is almost exactly the same as $|C_1|$. Therefore, as discussed before, it does not mean a large compression ratio is necessarily better. However, when $|\Sigma|' = 3$, $|C|$ is much smaller than $|C_1|$. It concurs with the analysis in Sect. 3. It is wise to pick a relatively small value of ϵ_1 and a relatively large value of ϵ_2. For example, when $(l, d) = (16, 3)$, setting $|\Sigma|' = 3$ and $n' = 4$ will reduce the size of candidate set from 203 to 7.

The running time of EMS2 for $(l, d) = (8, 1), (12, 2), (16, 3)$ are 0.14 s, 14.86 s and 21.18 m, respectively. Table 2 shows that generally EMS3 has a good speedup over EMS2. The best speedups for $(l, d) = (8, 1), (12, 2), (16, 3)$ are 1.75, 1.68 and 1.84 when $n' = 8$ (or 10), 4, 2. Figure 1 shows the speedups of EMS3 over EMS2 with different n' on the challenging instances. When n' is chosen appropriately, EMS3 is expected to have around 70% or 80% improvement in speed. When the alphabet size or the number of sequences is larger, EMS3 is expected to perform much better than EMS2. There will also be more choices for picking the parameters.

4.2 Real Biological Datasets

EMS3 is also compared against EMS2 on the real biological DNA datasets ($\Sigma = \{A, C, G, T\}$) discussed in [7,15]. The datasets can be downloaded from http://bio.cs.washington.edu/assessment/download.html.

Table 2. Running time of EMS3 with different n' ($|\Sigma|' = 3$)

n'	(l, d)		
	$(8, 1)$	$(12, 2)$	$(16, 3)$
2	5.83 s	1.24 m	**11.54 m**
4	0.72 s	**8.83 s**	12.70 m
6	0.15 s	10.15 s	14.61 m
8	**0.08 s**	11.67 s	17.22 m
10	**0.08 s**	13.45 s	19.87 m

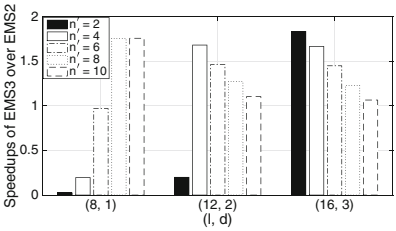

Fig. 1. Speedups of EMS3 over EMS2 with different (l, d) and n'

The "real" benchmark datasets (file names with suffix r) which have the binding sites in their real genomic promoter sequences are chosen as the test files. Datasets with less than 8 input sequences are excluded because they are not very challenging. For each dataset, d is set to 2 and 3. l is chosen on a dataset basis to ensure that the number of reported motifs is not excessive but the instance is challenging as well. When running the experiment of EMS3, instead of exhaustedly varying the parameters of ϵ_1 and ϵ_2, only one combination of ϵ_1 and ϵ_2 is tested. However, in the real datasets, no assumption should be made on the statistical distribution. Therefore, the second guideline can be utilized here to choose the parameters. For example, manually set $\epsilon_2 = 3/4$ and $\epsilon_1 = (1-\epsilon_1)\epsilon_2^d$ and solve it for $\epsilon_1 = 9/25$ when $d = 2$ and $\epsilon_1 = 27/91$ when $d = 3$.

In Table 3, the dataset name, the total number of sequences, the total number of bases in each dataset, the l and d combination, size of candidate motif set when running EMS3, the runtimes of the two algorithms and the speedup of EMS3 over EMS2 are reported. From this table, it is obvious that size of the candidate motif set is generally greatly reduced (*i.e.*, $|C| < |C_1|$). This shows the necessity and effectiveness of pruning $|C_1|$ by checking if the pattern in $|C_1|$, after encoding, can survive in $|C_2|$. When $d = 2$, the improvement in speed is around 30% to 60%, but in rare cases, it only performs slightly better than EMS2. It may be because for some small instances, the overhead brought by EMS3 more or less balances out its advantage. However, when $d = 3$, the improvement in speed is generally over 50% with the maximum speedup of 2.1 when $(l, d) = (17, 3)$ on mus11r dataset.

4.3 Summary of Experimental Evaluation

EMS3 outperforms EMS2 on both synthetic and real datasets. EMS3 works well with challenging instances. This is because the size of the candidate motif set is relatively small thus it will not take much time to verify the candidate motifs. If there is only one motif found which is the planted motif, then EMS3 is very good at capturing this one. EMS3 works better for large datasets and instances. It is expected EMS3 will perform better on protein data and datasets with more input sequences. This is proved in the time complexity analysis above. However, the corresponding experiments are not carried out because EMS2 consumes more than 500 GB memory for protein data even when $d = 3$.

Table 3. Running time of EMS3 over EMS2 on real datasets

Dataset	n	No. bases	l	d	$\|C_1\|$	$\|C\|$	T_{EMS3} (s)	T_{EMS2} (s)	Speedup
hm01r	18	36000	14	2	1	0	65.63	89.93	1.37
			18	3	0	0	5774.66	9104.86	1.58
hm02r	9	9000	15	2	68	7	19.44	23.58	1.21
			19	3	266059	1974	1260.21	2065.53	1.64
hm03r	10	15000	15	2	219	18	32.41	42.56	1.31
			19	3	196662	2079	2288.51	3931.03	1.72
hm04r	13	26000	14	2	349	102	44.45	61.27	1.38
			18	3	14938	2338	3232.39	6280.58	1.94
hm08r	15	7500	13	2	5	0	8.07	12.16	1.51
			17	3	1	0	570.29	1107.60	1.94
hm20r	35	70000	13	2	10	7	82.63	132.36	1.60
			17	3	0	0	8809.21	12406.11	1.41
hm26r	9	9000	15	2	220	51	17.32	23.60	1.36
			19	3	105908	579	1121.93	2152.78	1.92
mus02r	9	9000	15	2	178	75	17.40	23.85	1.37
			19	3	230537	3285	1455.39	2026.11	1.39
mus11r	12	6000	13	2	639	15	6.14	9.75	1.59
			17	3	3443	27	417.52	878.54	2.10
yst01r	9	9000	15	2	154	9	17.03	23.62	1.39
			19	3	213654	3920	1278.39	1969.11	1.54
yst03r	8	4000	14	2	8462	421	6.92	7.28	1.05
			19	3	30398	5	441.78	830.92	1.88
yst08r	11	11000	14	2	3603	510	20.91	23.96	1.15
			18	3	20359	831	1432.57	2093.48	1.46
yst09r	169	16000	13	2	96	66	17.17	27.89	1.62
			17	3	784	215	1560.99	2483.25	1.59

5 Conclusions and Future Work

In this paper, EMS3, an improved algorithm is proposed to efficiently solve the EMS problem. EMS3 is a non-recursive divide and conquer algorithm and uses the idea of projection. Theoretical analysis shows that EMS3 is even more competitive for large datasets and challenging instances. The experimental results reveal that EMS3 outperforms EMS2 which is the state-of-the-art algorithm.

In future the authors plan to improve the performance of EMS solvers by reducing the memory usage and focusing on larger datasets. Quorum support can be added to the existing EMS solver. How to project l, d or m in EMS

problems is worth considering as well. Besides, developing efficient approximate and randomized algorithms for the EMS problem is also interesting.

Acknowledgment. This work has been supported in part by the NSF grants 1447711, 1743418, and 1843025.

References

1. Adebiyi, E.F., Kaufmann, M.: Extracting common motifs under the levenshtein measure: theory and experimentation. In: Guigó, R., Gusfield, D. (eds.) WABI 2002. LNCS, vol. 2452, pp. 140–156. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45784-4_11
2. Buhler, J., Tompa, M.: Finding motifs using random projections. In: Proceedings of Fifth Annual International Conference on Computational Molecular Biology (RECOMB) (2001)
3. Cai, X., Mamun, A.A., Rajasekaran, S.: Novel algorithms for finding the closest l-mers in biological data. In: 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 525–528. IEEE (2017)
4. Cai, X., Zhou, S., Rajasekaran, S.: Jump: a fast deterministic algorithm to find the closest pair of subsequences. In: Proceedings of the 2018 SIAM International Conference on Data Mining, pp. 73–80. SIAM (2018)
5. Knuth, D.E., Morris Jr., J.H., Pratt, V.R.: Fast pattern matching in strings. SIAM J. Comput. **6**(2), 323–350 (1977)
6. Lanctot, J.K., Li, M., Ma, B., Wang, S., Zhang, L.: Distinguishing string selection problems. Inf. Comput. **185**(1), 41–55 (2003)
7. Nicolae, M., Rajasekaran, S.: Efficient sequential and parallel algorithms for planted motif search. BMC Bioinform. **15**(1), 1 (2014)
8. Nicolae, M., Rajasekaran, S.: qPMS9: an efficient algorithm for quorum planted motif search. Sci. Rep. **5**, 7813 (2015)
9. Pal, S., Xiao, P., Rajasekaran, S.: Efficient sequential and parallel algorithms for finding edit distance based motifs. BMC Genomics **17**(4), 465 (2016)
10. Pathak, S., Rajasekaran, S., Nicolae, M.: EMS1: an elegant algorithm for edit distance based motif search. Int. J. Found. Comput. Sci. **24**(04), 473–486 (2013)
11. Rajasekaran, S., et al.: High-performance exact algorithms for motif search. J. Clin. Monit. Comput. **19**(4–5), 319–328 (2005)
12. Rajasekaran, S., Dinh, H.: A speedup technique for (l, d)-motif finding algorithms. BMC Res. Notes **4**(1), 54 (2011)
13. Rocke, E., Tompa, M.: An algorithm for finding novel gapped motifs in DNA sequences. In: Proceedings of the Second Annual International Conference on Computational Molecular Biology, pp. 228–233. ACM (1998)
14. Sagot, M.-F.: Spelling approximate repeated or common motifs using a suffix tree. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998. LNCS, vol. 1380, pp. 374–390. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054337
15. Tompa, M., et al.: Assessing computational tools for the discovery of transcription factor binding sites. Nat. Biotechnol. **23**(1), 137 (2005)
16. Xiao, P., Pal, S., Rajasekaran, S.: qPMS10: a randomized algorithm for efficiently solving quorum Planted Motif Search problem. In: 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 670–675. IEEE (2016)
17. Xiao, P., Pal, S., Rajasekaran, S.: Randomised sequential and parallel algorithms for efficient quorum planted motif search. Int. J. Data Min. Bioinform. **18**(2), 105–124 (2017)