Computational Biology and Bioinformatics
2016; 4(2): 15-20
http://www.sciencepublishinggroup.com/j/cbb
doi: 10.11648/j.cbb.20160402.11
ISSN: 2330-8265 (Print); ISSN: 2330-8281 (Online)

A New Idea for Improving the Running Time of PMS Algorithm

Saeed Alirezanejad Gohardani[1], Mehri Bagherian[1, *], Hamid Reza Vaziri[2]

[1]Department of Applied Mathematics, Faculty of Mathematical Science, University of Guilan, Rasht, Iran

[2]Department of Biology, Faculty of Science, University of Guilan, Rasht, Iran

Email address:

sgohardani@phd.guilan.ac.ir (S. A. Gohardani), mbagherian@guilan.ac.ir (M. Bagherian), vaziri@guilan.ac.ir (H. R. Vaziri)

[*]Corresponding author

To cite this article:

Saeed Alirezanejad Gohardani, Mehri Bagherian, Hamid Reza Vaziri. A New Idea for Improving the Running Time of PMS Algorithm. *Computational Biology and Bioinformatics*. Vol. 4, No. 2, 2016, pp. 15-20. doi: 10.11648/j.cbb.20160402.11

Received: May 3, 2016; **Accepted**: May 16, 2016; **Published**: May 28, 2016

Abstract: Motif finding problem is a major challenge in biology with significant applications in the detection of transcription factor binding sites and transcriptional regulatory elements that are crucial in understanding gene expression and function, human disease, drug design, *etc*. Two type of motif finding problems have been investigated. Planted Motif Search Problem (PMSP) which is defined as finding motifs that appear in all sequences and a restricted version of it "Planted Motif Search Problem-Sample Driven" (PMSP-SD) where the motifs themselves are found in the input. The first version is NP-Complete and the second version can be trivially solved in polynomial time. In this paper, a new idea is used to speed up the PMS-SD algorithm. Although PMS-SD is a polynomial time algorithm and the new idea does not improve its asymptotic runtime, but since most of the motif search algorithms combine a sample driven approach with a pattern driven approach, the speed up of PMS-SD running time would result in speed up of PMS algorithm. To verify the performance of the modified algorithms which are called PMS-two step and PMS-SD-two step, these algorithms are tested on simulated data. The experimental results approve the improvements.

Keywords: Pattern and Motif Discovery, Planted (l,d)-Motif Search Problem, Closest Substring Problem

1. Introduction

Motifs which approximate conserved sequences across DNA/protein sequences, lead biologists to new biological discoveries. Regulatory regions in a genome such as promoters, enhancers, locus control regions contain motifs that control many biological processes such as gene expression [1] and [2]. In fact, some proteins known as transcription factors that bind to motif locations in regulatory regions can regulate gene expression.

A large number of methods have been proposed to investigate motifs in biological sequences. A class of approaches are combinatorial approaches such as *Planted Motif Search* (PMS), *Simple Motif Search* (SMS), and *Edited-distance-based Motif Search* (EMS) [3].

Among different versions of combinatorial approaches, PMS problem is more popular due to its closeness to motif reality. Motif in PMS problem is referred as a *(l,d)-motif,* where l is the length of the motif and d is the number of mismatches allowed for its instances. This problem is trying to extract common substrings that appear in every input sequence with pre-specified mismatches allowed. In fact, these instances are *(l,d)-motifs* which has length l and allowed to have d mismatches in different places in each of them. An algorithm that solves PMS problem is called *PMS Algorithm*. Motif finding problem is based on two categories, Sample-Driven and Pattern-Driven approaches. Using pattern-Driven approaches, one tries all possible $|\Sigma|^l$ *l-mers* as motif candidates which is an exponential search space, but in Sample-Driven approaches, all possible motifs generated from the *l-mers* in input strings are of interest which could be found in polynomial time. It has been proven that PMS problem is NP-hard which means unlikely any algorithm solves it in polynomial time [4]. Du to NP-hardness, two kinds of PMS algorithms are *exact* and *approximate* algorithms. An exact algorithm can find all the motifs in input sequences, while an

approximate algorithm may not be able to find all of them. All existing exact algorithms solve PMS problem in exponential time in some of its parameters. Some of the most important exact algorithms are PMS1 [5], PMS2 [5], PMS3 [5], PMSi [6], PMSP [6], PMSP4 [7], Stemming [8], PMS5 [9], PMS6 [10], PMS8 [11], qPMS9 [12], PMS Prune [13], Algorithm Voting [14] and RISSOTO [15]. On the other side, approximate algorithms take less time than exact algorithms. They usually employ heuristics such as local search, Gibbs sampling, exponential optimization, *etc*. Some examples of approximate algorithms are Algorithm MEME [16], Algorithm PROJECTION [17], Algorithm Gibbs DNA [18], Algorithm WINNOWER [19], and Algorithm Random Projection [20]. Some other approximate PMS algorithms are MULTIPROFILER [21], Algorithm Pattern Branching [22], Algorithm Profile Branching [22], Algorithm CONSENSUS [23] and genetic algorithm [24].

In this paper a new simple idea is proposed to speed up the PMS-SD algorithm and since PMS algorithms use PMS-SD as subroutines the faster PMS-SD algorithm results in speed up of PMS algorithm. The remainder of this paper is organized as follows. In section2, some definitions and theorems are introduced. In section 3, Algorithm PMS will be described briefly, and then a new algorithm is proposed based on PMS. Then, in section 4 PMS-SD algorithm in which the searched zone is restricted to input sequences is improved. Experimental results are shown in section 5 and section 6 ends the paper with conclusion.

2. Definitions and Theorems

Definition 1. Let x and x^+ be two strings with length l over an alphabet Σ. We show them as $x = x_1, x_2, \dots, x_l$ and $x^+ = x_1^+, x_2^+, \dots, x_l^+$. x^+ is called an *1-right neighbour* of x if $x_1^+ = x_2, x_2^+ = x_3, \dots, x_{l-1}^+ = x_l$.

Definition 2. Let s be a string with length n and x be a string with length l (l<n). x is called an *l-mer* of s and is shown with $x \in_l s$ if x is a contiguous substring of s.

Theorem 1. [19] Let s_i and s_j be two strings over an alphabet Σ, $x \in_l s_i$ and $y \in_l s_j$. Let $d_H(x,y) = d$, where d_H is the Hamming distance between two strings with equal length, i.e., the number of positions where the two strings differ. Then

$$d_H(x^+, y^+) = d - 1 \text{ or } d \text{ or } d + 1$$

Figure 1. shows two strings s_i and s_j with two contiguous l-mers in each one.

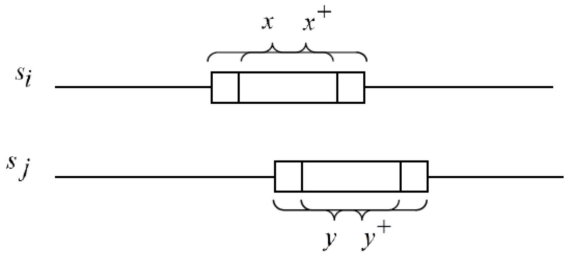

Figure 1. *Two strings s_i and s_j with two contiguous l-mers in each one.*

Theorem 2. Let s_i and s_j be two strings over an alphabet Σ and $x \in_l s_i$ and $y \in_l s_j$ and $d_H(x,y) = d$ then

1. If $x_1 \neq y_1$ and $x_l^+ = y_l^+$ then $d_H(x^+, y^+) = d - 1$.
2. If $x_1 \neq y_1$ and $x_l^+ \neq y_l^+$ then $d_H(x^+, y^+) = d$.
3. If $x_1 = y_1$ and $x_l^+ = y_l^+$ then $d_H(x^+, y^+) = d$.
4. If $x_1 = y_1$ and $x_l^+ = y_l^+$ then $d_H(x^+, y^+) = d + 1$.

Proof. $d_H(x^+, y^+) = d_H(x_2x_3 \dots x_l x_l^+, y_2y_3 \dots y_l y_l^+ = d_H(x_2x_3 \dots x_l, y_2y_3 \dots y_l) + d_H(x_l^+, y_l^+)$

If $x_1 \neq y_1$ and $x_l^+ = y_l^+$ then $d_H(x^+, y^+) = (d-1) + 0 = d - 1$

(2), (3) and (4) would be proven in exactly the same way as (1).

Definition 3. Let s and x be two strings with lengths n and l, such that l<n, respectively. We define

$$c_d(x,s) = \{y \in_l s : d_H(x,y) \leq d\}$$

$$B_d(x,s) = \{y \in_l s : d_H(x,y) = d\}$$

$$B_d(x) = \{y: |y| = 1, d_H(x,y) = d\}$$

Based on theorem 2, if $x \in_l s$ and $d_H(x,y) \leq d + 1$ then y^+ has a chance to be in $c_d(x^+, s)$. The next theorem is based on this fact.

Theorem 3. Let s_i and s_j be two strings over an alphabet Σ.

1. If $d_H(x,y) < d$ then $y^+ \in c_d(x^+, s_j)$.
2. If $d_H(x,y) = d$ and $x_1 \neq y_1$ then $y^+ \in c_d(x^+, s_j)$.
3. If $d_H(x,y) = d$, $x_1 = y_1$ and $x_l^+ = y_l^+$ then $y^+ \in c_d(x^+, s_j)$.
4. If $d_H(x,y) = d + 1, x_1 \neq y_1$ and $x_l^+ = y_l^+$ then $y^+ \in c_d(x^+, s_j)$.

Proof. If $d_H(x,y) < d$ then using theorem1 we have $d_H(x^+, y^+) \leq d$ thus $y^+ \in c_d(x^+, s_j)$. Using theorem1 and theorem 2, it will be trivial to prove (2), (3), and (4).

Corollary1 enables us to reduce the number of comparisons when $d_H(x,y) = d$.

Corollary 1. Let s_i and s_j be two strings over an alphabet Σ.

1. If $d_H(x,y) < d$ then $y^+ \in c_d(x^+, s_j)$.
2. If $d_H(x,y) = d$, $x_1 = y_1$ and $x_l^+ \neq y_l^+$ then $y^+ \notin c_d(x^+, s_j)$.
3. If $d_H(x,y) = d + 1$ $x_1 \neq y_1$ and $x_l^+ = y_l^+$ then $y^+ \in c_d(x^+, s_j)$.

Proof. (1) and (3) are similar to (1) and (4) in theorem3. (2) is exactly opposite of theorem3's (2) and (3).

Theorem1 and definition of $B_d(x,s)$ help us to reduces local search in input sequences which is a part of all motif search algorithms. This result would be proposed in the next theorem.

Theorem 4. Let s_i and s_j be two strings over an alphabet Σ.

1. If $d_H(x,y) = d - 1$ $x_1 = y_1$ and $x_l^+ \neq y_l^+$ then $y^+ \in B_d(x^+, s_j)$.
2. If $d_H(x,y) = d$ $x_1 \neq y_1$ and $x_l^+ \neq y_l^+$ then $y^+ \in B_d(x^+, s_j)$.

3. If $d_H(x,y)=d$ $x_1=y_1$ and $x_l^+=y_l^+$ then $y^+ \in B_d(x^+,s_j)$.
4. If $d_H(x,y)=d+1$ $x_1\neq y_1$ and $x_l^+=y_l^+$ then $y^+ \in B_d(x^+,s_j)$.

Proof. As we know

$$d_H(x^+,y^+)=d_H(x_2x_3\dots x_lx_l^+,y_2y_3\dots y_ly_l^+)$$

$$=d_H(x_2x_3\dots x_l,y_2y_3\dots y_l)+d_H(x_l^+,y_l^+)$$

If $d_H(x,y)=d-1$ $x_1=y_1$ and $x_l^+\neq y_l^+$ then we have $d_H(x^+,y^+)=(d-1)+1=d$ thus $y^+\in B_d(x^+,s_j)$.

(2), (3) and (4) can be proven in exactly the same way as (1).

Definition 4 (planted motif search problem). Let $\{s_i\}_{i=1}^t$ is a set of strings with length n over an alphabet Σ and nonnegative integers l, d, satisfying $0\le d<l<n$. The *(l,d)-motif search problem* is to find a string x, called *motif*, of length l such that

$$\forall i=1,\dots,t\ \exists z\in_l s_i:\ d_H(x,z)=d$$

Now, we use corrolary1 and theorem4 to propose a technique for speed up the PMS algorithm which we call it PMS two-step.

3. PMS Two-Step Algorithm for Pattern-Driven Motif

First of all, we describe Algorithm PMS briefly, because our new idea is applied to it. For more details about Algorithm PMS, the reader is referred to [13].

The main trend of the algorithm is as follows. First for each *l-mer* x of the first string, find all of its neighbors with distance d. Each of these neighbors could be a motif candidate. Let x' be one of these neighbors. To investigate whether x' could be a motif, we do as follows. If there is an *l-mer* with distance d from x' in each string that is in distance $2d$ from x, then x' is introduced as an instance of motif. Algorithm PMS could be presented as the following pseudocode.

Algorithm PMS
Input: $S=\{s_1,s_2,\dots,s_n\}$
Output: A set M of candidate motifs

$M=\phi$

For each x, $x\in_l s_1$:
Construct $c_{2d}(x,s_j)$ for $j=2,\dots,t$
For each $x'\in B_d(x)$
If for each j, $(2\le j\le t)$ $z_j\in c_{2d}(x,s_j)$ exists such that $d_H(x',z_j)=d$
Add x' to M.
Return M.

Theorem 6 [5]. Algorithm PMS can be implemented in $o\left(tn^2\binom{l}{d}|\Sigma|^d\right)$ time and in $o(tn^2)$ space.

We use corollary1 to propose *Algorithm PMS two-step*.

Algorithm PMS two-step is similar to *PMS* for odd *l-mers* of the first string, but for even *l-mers* there is no need to construct $c_{2d}(x,s_j)$ for $j=2,\dots,t$ since we can extract it from $c_{2d}(x,s_j)$ which was calculated for odd *l*-mers at previous step, according to corollary1. The pseudo code of the *Algorithm PMS two-step is as follows.*

Algorithm PMS two-step
Input: $S=\{s_1,s_2,\dots,s_n\}$
Output: A set M of candidate motifs
$M=\phi$

For each x, $x\in_l s_1$ that started from odd positions of s_1 ($n-l$ must be odd):
Construct $B_{2d}(x,s_j)$ and $B_{2d+1}(x,s_j)$ for j=2,...,t.
For each $x'\in_l B_d(x)$
If for each j, $(2\le j\le t)$ there exists $z_j\in c_{2d}(x,s_j)$ such that $d_H(x',z_j)=d$
Add x' to M.
If $y\in c_{2d}(x,s_j)$ and is started from position *n-l+1* of s_j then delete it from $c_{2d}(x,s_j)$ for j=2,…,t.
For each $y\in c_{2d}(x,s_j)$ add y^+ to $c_{2d}(x^+,s_j)$ for j=2,…,t.
For each $y\in B_{2d}(x,s_j)$ for $j=2,\dots,t$
If $x_1=y_1$ and $x_l^+\neq y_l^+$ then delete y^+ from $c_{2d}(x^+,s_j)$.
For each $y\in B_{2d+1}(x,s_j)$ for j=2,…,t
If $x_1\neq y_1$ and $x_l^+=y_l^+$ then add y^+ to $c_{2d}(x^+,s_j)$.
For $y\in_l s_j$ which started from position 1 of s_j if $d_H(x^+,y)\le 2d$ add y to $c_{2d}(x^+,s_j)$ for j=2,…,t.
For each $x'\in B_d(x^+)$
If for each j, $(2\le j\le t)$ there exists $z_j\in c_{2d}(x^+,s_j)$ such that $d_H(x',z_j)=d$
Add x' to M.
Return M

Steps 1 to 6 of this algorithm are similar to *PMS* algorithm. In steps 7 to 13 $c_{2d}(x^+,s_j)$ is constructed for $x^+\in_l s_1$ according to corollary 1. The key step of Algorithm PMS two-step is step7 in which *n-l+1* position of string s_j for $j=2,\dots,t$ are omitted from $c_{2d}(x,s_j)$. On the other hand, since the elements of $c_{2d}(x,s_j)$ are ordered, if element *n-l+1* exists, it is the last element of $c_{2d}(x,s_j)$.

Steps 8 to 12 are according to corollary 1. Step 13 is based on the fact that the first *l-mers* of strings 2 to t cannot be obtained based on corollary 1. So we have to investigate these *t-1* remained *l-mers* for *2d* distances.

After constructing $c_{2d}(x^+,s_j)$ in steps 7 to 13, candidate motifs are investigated for x^+regarding the new $c_{2d}(x^+,s_j)$. Steps 14 to 16 are similar to steps 4 to 6.

Theorem 8. The *PMS two-step* algorithm can be implemented in $o\left(tn^2l\binom{l}{d}|\Sigma|^d\right)$ time and in $o(tn^2)$ space.

Proof. The PMS two-step algorithm implements in $\frac{n}{2}\left(tnl+2\binom{l}{d}|\Sigma|^d tnl+2tn+tl\right)$ time and therefore its running time is $o\left(tn^2l\binom{l}{d}|\Sigma|^d\right)$. On the other hand, this algorithm uses $B_{2d}(x,s)$, $B_{2d+1}(x,s), c_{2d}(x,s)$ and $c_{2d}(x^+,s_j)$ space. Our algorithm uses $o(4tn^2)=o(tn^2)$

space to does this process.

If $x' \in B_d(x)$ is changed to $x' \in B_d(x, s_k)$ for *k=1,2,...,t* PMS algorithm is restricted to find motifs which are present in input sequences and is called PMS-SD algorithm.

In this section, we use corollary 1 and theorem 4 to improve PMS algorithm for sample-driven motif search problem. In other words, we search motifs which are present in the given strings.

Algorithm PMS-SD two-step
Input: $S = \{s_1, s_2, \dots, s_n\}$
Output: A set M of candidate motifs
$M = \phi$

For each *x*,$x \in_l s_1$ that started from odd positions of s_1 (*n-l* must be odd)

Construct $c_{2d}(x, s_j)$, $B_{2d}(x, s_j)$ and $B_{2d+1}(x, s_j)$ for *j=2,..,t*.

Construct $B_d(x, s_j)$, $B_{d-1}(x, s_j)$ and $B_{d+1}(x, s_j)$ for *j=2,..,t*.

For each $x' \in B_d(x, s_k)$ for *k=1,...,t*

If for each j $(2 \le j \le t)$ there exists $z_j \in c_{2d}(x, s_j)$ such that $d_H(x', z_j) = d$

Add x' to M.

If $y \in c_{2d}(x, s_j)$ and started from position *n-l+1* of s_j delete it from $c_{2d}(x, s_j)$ for *j=2,...,t*.

If $y \in B_d(x, s_j)$ is started from position *n-l+1* of s_j delete it from $B_d(x, s_j)$ for *j=1,...,t*.

For each $y \in c_{2d}(x, s_j)$ add y^+ to $c_{2d}(x^+, s_j)$ for *j=2,...,t*.

For each $y \in B_{2d}(x, s_j)$ for *j=2,...,t*

If $x_1 = y_1$ and $x_l^+ \neq y_l^+$ then delete y^+ from $c_{2d}(x^+, s_j)$.

For each $y \in B_{2d+1}(x, s_j)$ for *j=2,...,t*

If $x_1 \neq y_1$ and $x_l^+ = y_l^+$ then add y^+ to $c_{2d}(x^+, s_j)$.

For each $y \in B_{d-1}(x, s_j)$ for *j=2,...,t*

If $x_1 = y_1$ and $x_l^+ \neq y_l^+$ then add y^+ to $B_d(x^+, s_j)$.

For each $y \in B_d(x, s_j)$ for *j=2,...,t*

If $x_1 \neq y_1$ and $x_l^+ \neq y_l^+$ then add y^+ to $B_d(x^+, s_j)$.

If $x_1 = y_1$ and $x_l^+ = y_l^+$ then add y^+ to $B_d(x^+, s_j)$.

For each $y \in B_{d+1}(x, s_j)$ for *j=2,...,t*

If $x_1 \neq y_1$ and $x_l^+ = y_l^+$ then add y^+ to $B_d(x^+, s_j)$

For $y \in_l s_1$ that started from position 1 of s_1 if $d_H(x^+, y) = d$ add *y* to $B_d(x^+, s_1)$

For each $y \in_l s_j$ that started from position 1 of s_j for *j=2,...,t*

If $d_H(x^+, y) = d$ add *y* to $B_d(x^+, s_j)$ and $c_{2d}(x^+, s_j)$

Else if $d_H(x^+, y) \le 2d$ add *y* to $c_{2d}(x^+, s_j)$

For each $x' \in B_d(x^+, s_k)$ for *k=1,...,t*

If for each j $(2 \le j \le t)$ if $z_j \in c_{2d}(x^+, s_j)$ exists such that $d_H(x', z_j) = d$

Add x' to M

Return M

Steps 1 to 7 are similar to PMS algorithm. In steps 8 to 25 the two sets $c_{2d}(x^+, s_j)$ and $B_d(x^+, s_j)$ are constructed. In steps 8 and 9 positions *n-l+1* are omitted from s_j. In steps 10 to 14 and 15 to 21 $c_{2d}(x^+, s_j)$ and $B_d(x^+, s_j)$ are constructed according to corollary 1 and theorem 4, respectively. In steps 22 to 25 the construction process of $c_{2d}(x^+, s_j)$ and $B_d(x^+, s_j)$ are completed by calculating the distance between the first t sequences of these strings and x^+. Finally, the motif search process among motif candidates would be completed in steps 26 to 28.

Theorem 9. PMS-SD two-step algorithm can be implemented in $o\left(tn^2 l \binom{l}{d} |\Sigma|^d\right)$ time and in $o(tn^2)$ space.

Proof. The PMS two-step algorithm for sample-Driven motif implements in $\frac{n}{2}\left(tnl + 2\binom{l}{d}|\Sigma|^d tnl + 2tn + 2tn\right)$ time and therefore we have $o\left(tn^2 l \binom{l}{d} |\Sigma|^d\right)$ time. On the other hand, this algorithm uses sets $B_{2d}(x, s)$, $B_{2d+1}(x, s)$, $B_{d-1}(x, s)$, $B_{d+1}(x, s)$, $c_{2d}(x, s_j)$ and $c_{2d}(x^+, s_j)$ as space. Our algorithm uses $o(6tn^2) = o(tn^2)$ space to does this process.

4. Experimental Results

4.1. Finding the Simulated DNA Motif

The PMS-SD two-step and PMS two-step algorithms are evaluated by computational experiments. To do this, a set of random data is generated according to what described in [13]. First of all, 20 strings of length 600 are generated randomly such that each letter has the equal probability $1/|\Sigma|$. Then, a motif of length *l* is generated randomly in the same manner. Next, 20 instances are generated from the motif by the mutating the letter at exactly *d* random positions. Finally, they are planted in each sequence where each position is selected randomly.

The proposed algorithms have been written in C. The computational comparison has been performed in a laptop computer with an Intel Core i3-M330 CPU (2.13 GH) and 4 GB memory. PMS and PMS two-step's running time have been reported in table 1 and Algorithm PMS-SD and Algorithm PMS-SD two-step in table 2. Based on Table 2, PMS-SD two-step has reduced PMS-SD's running time for about 40 percent.

Table 1. *Time comparison of PMS, PMS two-step and voting algorithms for DNA sequences.*

(*l*,*d*)	PMS	PMS two-step	Voting [14]
(11,3)	4.6s	4.4s	5.7s
(13,4)	102.1s	93.5s	72.5s
(15,5)	23.5m	22.7m	14.8m
(17,6)	8h	7.8h	*

'*' denotes that its time was not reported.

Table 2. *Time comparison of Algorithm PMS-SD and Algorithm PMS-SD two-step for DNA sequences.*

(l,d)	PMS-SD	PMS-SD two-step
(9,2)	1.03s	0.76s
(11,3)	1.2s	0.89s
(13,4)	1.37s	1.03s
(15,5)	1.58s	1.13s
(17,6)	1.71s	1.23s
(19,7)	1.83s	1.29s
(21,8)	2s	1.37s

(l,d)	PMS-SD	PMS-SD two-step
(23,9)	2.17s	1.43s
(25,10)	2.26s	1.5s
(29,12)	2.55s	1.63s
(29,13)	3.01s	2.04s
(29,14)	4.95s	4.06s
(51,23)	3,54s	2s
(51,24)	3.56s	2.03s
(51,25)	3.59s	2.08s

4.2. Finding Real DNA Motif

The proposed algorithm tested with the real DNA data sets, preproinsulin, DHFR, metallothionein, c-fos and Yeast ECB data sets as in [25]. Algorithm PMS two-step found transcription regulatory elements on these data set.

5. Conclusion

In this paper, a new idea was used to speed up the PMS algorithm's running time. Using some characteristics of DNA strings, some theorems has been proven to reduce the number of calculations of distances between two strings with the same length. Algorithms PMS two-step and PMS-SD two-step based on Algorithm PMS were proposed. As Table 1 shows, although the voting algorithm finds the (15,5) motif in 14.8 m which is better than PMS two-step (22.7 m) and PMS (23.5 m), this algorithm fails to find (17,6) motif which is found by PMS two-step (8 h) and PMS (7.8 h). For SD motif search problem Table 2 shows the results obtained from Algorithm PMS-SD and Algorithm PMS-SD two-step. As it is seen for all tested (l,d) motifs, the running time of Algorithm PMS-SD two-step is better than Algorithm PMS-SD. For example for (25,10) motif Algorithm PMS-SD two-step serves 1.5 s while Algorithm PMS-SD needs 2.26 s to search the motif.

Acknowledgments

This work is supported by a grant from university of Guilan. The authors appreciate the valuable comments provided by the anonymous referees which lead to this improved version of the paper.

References

[1] D. Laurent and B. Philipp, "Searching for regulatory elements in human noncoding sequences," *Current Opinion in Structural Biology,* vol. 7, pp. 399-406, 1997.

[2] D. S. a. D. I. Ratne, "Use of a Probabilistic Motif Search to Identify Histidine Phosphotransfer Domain-Containing Proteins," *PLOS one,* pp. 1-18, 2016.

[3] S. Rajasekaran, "Computational techniques for motif search," *Frontiers in Bioscience,* no. 14, pp. 5052-5065, 2009.

[4] M. Frances and A. Litman, "On Covering Problems of Codes," *Theory of Computing Systems,* vol. 30, no. 2, pp. 113-119, 1997.

[5] S. Rajasekaran, S. Balla and C. Huang, "EXACT ALGORITHMS FOR PLANTED MOTIF CHALLENGE PROBLEMS," *APBC,* pp. p249-259, 2005.

[6] J. Davila, S. Balla and S. Rajasekaran, "Space and Time Efficient Algorithms for Planted," *Proc. Second Int Workshop Bioinformatics Research and Applications,* May 2006.

[7] S. Rajasekaran and H. Dinh, "A Speedup Technique for (l, d) Motif Finding Algorithms," *BMC Research Notes,* vol. 4, no. 54, Mar. 2011.

[8] P. Kuksa and V. Pavlovic, "Efficient Motif Finding Algorithms for Large-Alphabet Inputs," *BMC Bioinformatics,* vol. 11, May 2010.

[9] H. Dinh, S. Rajasekaran and V. Kundeti, "PMS5: An Efficient Exact Algorithm for the (l, d) Motif Finding Problem," *BMC Bioinformatics,* vol. 12, Oct. 2011.

[10] S. Bandyopadhyay, S. Sahni and S. Rajasekaran, "PMS6: A Fast Algorithm for Motif Discovery," in *Proc. IEEE Second Int'l Conf. Computational Advances in Bio and Medical Sciences (ICCABS '12)*, Feb. 2012.

[11] M. N. a. S. Rajasekaran, "Efficient sequential and parallel algorithms for planted motif search," *BMC Bioinformatics,* pp. DOI: 10.1186/1471-2105-15-34, 2014.

[12] M. N. a. S. Rajasekaran, "qPMS9: An Efficient Algorithm for Quorum Planted Motif Search," *Scientific Reports,* p. doi: 10.1038/srep07813, 2015.

[13] Davila, J.; Balla, S.; Rajasekaran, S., "Fast and Practical Algorithms for Planted (l,d) Motif Search," *IEEE/ACM Trans. on Computational Biology and Bioinformatics,* vol. 4, no. 4, pp. 544-552, 2007.

[14] F. Chin and H. Leung, "Voting Algorithms for discovering long motifs," in *Proceedings of the Third Asia-Pacific Bioinformatics Conference (APBC2005)*, Singapore, pages 261-271, 2005.

[15] N. Pisanti, A. Carvalho, L. Marsan and M. Sagot, "Risotto: Fast extraction of motifs with mismatches," in *Proceedings of the 7th Latin American Theoretical Informatics Symposium*, 2006.

[16] T. Baily and C. Elkan, "Fitting a mixture model by expectation maximization to discover motifs in biopolymers," in *Proceedings of Second International Conference on Intelligent Systems for Molecular Biology*, 1994.

[17] J. Buhler and M. Tompa, "Finding motifs using random projections," *Proceedings of Fifth Annual International Conference on Computational Molecular Biology (RECOMB),* pp. 69-76, 2001.

[18] C. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald and e. al, "Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment," *Science,* vol. 262, pp. 208-214, 1993.

[19] P. Pevzner and S. Sze, "Combinatorial approaches to finding subtle signals in dna sequences," *Proceedings of Eighth International Conference on Intelligent Systems for Molecular Biology,* pp. 269-278, 2000.

[20] E. Rocke and M. Tompa, "An algorithm for finding novel gapped motifs in dna sequences," *Proceedings of Second International Conference on Computational Molecular Biology (RECOMB),* pp. 228-233, 1998.

[21] U. Keich and P. Pevzner, "Finding motifs in the twilight zone," *Bioinformatics,* vol. 18, pp. 1374-1381, 2002.

[22] A. Price, S. Ramabhadran and P. Pevzner, "Finding subtle motifs by branching from sample strings," *Bioinformatics,* vol. 1, pp. 1-7, 2003.

[23] G. Hertz and G. Stormo, "Identifying dna and protein patterns with statistically significant alignments of multiple sequences," *Bioinformatics,* vol. 15, pp. 563-577, 1999.

[24] A. M. J. L. A. a. A. K. Joan Serr, "A Genetic Algorithm to Discover Flexible Motifs with Support," *arXiv:1511.04986v2 [cs.LG],* pp. 1-9, 2016.

[25] M. Blanchette, "Algorithms for phylogenetic Footprinting," *Proc. Fifth Ann. Int'l Conf. Computational Molecular Biology (RECOMB01),* Apr. 2001.

[26] J. Davila, S. Balla and S. Rajasekaran, "Pampa: An improved branch and bound algorithm for planted (l, d) motif search," *Technical report,* 2007.