An Algorithm to Find All Identical Motifs in Multiple Biological Sequences

Ashish Kishor Bindal[1], R. Sabarinathan[1], J. Sridhar[2], D. Sherlin[1], and K. Sekar[1,⋆]

[1] Bioinformatics Centre (Centre of excellence in Structural Biology and Bio-computing), Indian Institute of Science, Bangalore 560012, India
Tel.: +91-080-22933059/23601409; Fax: +91-080-23600683/23600551
[2] Center of Excellence in Bioinformatics, School of Biotechnology, Madurai Kamaraj University, Madurai 625021, Tamilnadu, India
sekar@physics.iisc.ernet.in, akbindal@ug.iiita.ac.in, sabari.binc@gmail.com, srimicro2002@gmail.com, sherlinsugirtha@gmail.com,
http://www.physics.iisc.ernet.in/~dichome/sekhome/index.html

Abstract. Sequence motifs are of greater biological importance in nucleotide and protein sequences. The conserved occurrence of identical motifs represents the functional significance and helps to classify the biological sequences. In this paper, a new algorithm is proposed to find all identical motifs in multiple nucleotide or protein sequences. The proposed algorithm uses the concept of dynamic programming. The application of this algorithm includes the identification of (a) conserved identical sequence motifs and (b) identical or direct repeat sequence motifs across multiple biological sequences (nucleotide or protein sequences). Further, the proposed algorithm facilitates the analysis of comparative internal sequence repeats for the evolutionary studies which helps to derive the phylogenetic relationships from the distribution of repeats.

Keywords: Sequence motifs, nucleotide and protein sequences, identical motifs, dynamic programming, direct repeat and phylogenetic relationships.

1 Introduction

A conserved pattern of a nucleotide or amino acid sequence with a specific biological function is known as a sequence motif and is becoming increasingly important in the analysis of gene regulations [1]. Research on protein and DNA sequences revealed that specific sequence motifs in biological sequences exhibit important characteristics [2]. In DNA sequences, the sequence motif act as specific binding sites for proteins (nuclease, transcription factors, etc.) and RNAs (mRNA splicing, transcription termination, etc.) [1]. Further in proteins, these motifs act as enzyme catalytic sites, prosthetic group attachment sites (haem, pyridoxal phosphate, biotin, etc.), metal binding amino acids, cysteines involved

⋆ Corresponding author.

in disulfide bonds or regions involved in binding a molecule [3]. In the recent years, due to the exponential rise in the volume of nucleotide and amino acid sequences in their respective databases, identification of sequence motifs using experimental methods is impossible. In addition, many newly discovered protein sequences do not share a global sequence similarity with a known protein. However, they share a short stretch of conserved sequences which represent the characteristics of similar domains [4]. Over the past years, these problems have been addressed using newly developed computational methods [5],[6],[7]. To this end, an efficient algorithm is proposed using the dynamic programming.

Earlier studies indicate that the transcription factor (TF) binding sites are well conserved motifs of short DNA sequence stretch. The motif size ranges from 5 to 35 nucleotides long and occur in a well-ordered and regularly spaced manner [8],[9]. For example, in eukaryotes the cis-regulatory module (CRMs) usually occurs in a fixed arrangement and distributed over very large distances. Further, the repeat occurrence of this binding site will help for the alternate modes of binding by the same protein which leads to the regulation of transcriptional activity. Gene duplications and recombination events are thought to be responsible for this repeat occurrence of sequence motifs. The distribution of repeats in archaea indicates that they have an intermediate relationship between prokaryotes and eukaryotes [10]. In DNA, these repeats are mainly classified into two groups such as tandem and interspersed repeats. The tandem repeats are an array of consecutive repeats and often associated with disease syndromes [11]. On the other hand, interspersed repeats are copies of transposable elements located at various regions in a genome. Moreover, the repeats that are separated by intermediate sequences of constant length occurring in clusters are referred to short regularly spaced repeats (SPSRs) [12]. Generally, these short repeats indicate the position of deletion and precise removal of transposable elements [13], where as, longer identical repeats are responsible for class switching in immunoglobulins [14]. Further, tandem repeats in telomers are involved in the protection of chromosome end and its length. In some cases, the internal sequence repeats in proteins adopt similar three-dimensional structures [15],[16]. However, further work is necessary to ascertain this aspect. In addition, the internal sequence repeats are observed to be associated with structural motifs or domains in the class of repeat protein families [17]. Further, the repeated sequence motifs play an active role in protein and nucleotide stability, thus, not only ensuring proper functioning [18] but some times cause malfunction and disease [19],[20].

1.1 Existing Algorithms

In the post genomic era, many algorithms are available in the literature to find the sequence motifs and repeats in biological sequences. However, these algorithms significantly vary in their methodologies. In general, the motif finding algorithms are divided into two major groups based on their working principle. The first group of algorithms identifies the motifs with reference to the annotated motif database. For example, the programs InterProScan [5], Motif Scan [21], ScanProsite [22] and SMART [23] search for motifs against protein profile database such as Prosite, Pfam, TMHMM etc. In addition, the above mentioned programs are limited to only protein sequences. Further, the program MOTIF [24] identifies the motif in both protein and DNA sequences using the above profile databases as well as user defined libraries. In contrast, another set of programs such as MEME [6], TEIRESIAS [7], ALIGN ACE [25], DILIMOT [26] and Gibbs Sampler [27] identify the motifs without any reference database. However, they use some statistical methods to identify the motifs and represent the conserved regions of the motifs in the form of sequence patterns using regular expressions or sequence logos. It is to note that most of these algorithms lack in the limitation of input sequence size (TEIRESIAS and ALIGN ACE take around 3,50,000 residues and the program MEME limits only to 60,000 residues).

The proposed algorithm has been developed by keeping the above lacuna in mind and uses the dynamic programming method implemented earlier [28] to identify all identical motifs present in multiple biological sequences (nucleotides and protein). To the best knowledge of authors, there is no such algorithm exists in the open literature. The proposed algorithm can be effectively used for the comparative identification of direct repeat motifs in several biological sequences. However, inorder to reduce the computational time, the total number of residues for a single run is restricted to a maximum of 10,00,000 residues.

2 Methodology

The proposed algorithm identifies all motifs which are present in a given set of biological sequences. Since, the problem of finding identical motifs in multiple sequences is similar to the problem of finding identical internal repeats in a sequence, when all sequences are concatenated with a delimiter or special character (z), where z $\notin \sum$ ($\sum$ represents a set of alphabet characters in the input sequences). The criteria for the identical motif should be an exact pattern repeated more than one sequence. Thus, we will refer the identical motif as identical repeat in the following sections. The algorithm adopts the methodology of FAIR algorithm [28]. In addition, it has been improved by using hash table to reduce the time complexity. The working principle of the new methodology is explained in the subsequent sections.

2.1 Pre-processing Phase

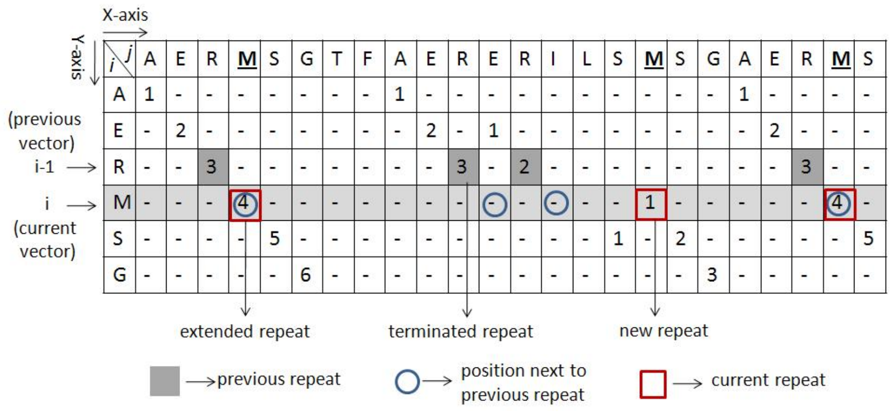
Initially, the uploaded sequences are concatenated with a delimiter at the end of each sequence and stored in a string S. In addition, the starting position of each input sequence in the string S is stored in an array. Further, a hash table is created to improve the execution time during search phase and to store the positions or occurrences of each alphabet (X) in the string S. The size of the hash table is equal to the length of the string S. The number of entries in the hash table varies for DNA (only four A,T,G and C) and protein (20 amino acids) sequences. All the positions of a single character (X) present in the string S are stored in a hash element or key (hash[X]). For example, the hash[X] represents the hash key of character X, the vector Voccurence[hash[X]] contains the occurences or positions of character X in string S and referred as Voccurence[X].

2.2 Searching Phase

The proposed algorithm uses the dynamic programming method to determine the identical repeats in the string S. The string S is aligned itself by taking the same on both X- and Y-axes in a two-dimensional space (see Fig. 1). Instead of creating a two-dimensional matrix for storing the match score values, the algorithm uses the concept of linear space complexity deployed in FAIR algorithm [28] by using two vectors (current and previous). The size of the current and previous vectors is equal to N (length of the string S). While scanning, each element (S[i]) in the Y-axis is used as a probe to search for the match along X-axis (S[j]), where i,j $\in$ 0 $\leq$i$\leq$N, 0$\leq$j$\leq$N (see Fig. 1). During this process, when an element S[i] from Y-axis is matched identically with the element S[j] of X-axis, a hit value of one is added to the value of $j - 1^{th}$ in the previous vector and the total is assigned to the j^{th} position of current vector. Thus, the current vector holds the present repeat length with respect to the character S[i] and the previous vector holds the repeat diagonal up to S[i-1]. Whenever a match is not found or the sequence ends (j==N), the value of the previous vector is checked for the size greater than the minimum length of the motif, then the previous vector value is stored as the length of the repeat (L) and the positions of i-1 and j-1 are stored as repeat end positions ($R_{i-1,j-1,L}$). The above operation is repeated recursively till the end of i along Y-axis. The pseudo-code for the recursive operation is given below,

```
IF S[i] equals to S[j] THEN
    set current[j] to previous[j-1]+1;
ENDIF
ELSE
 IF previous[j-1]>=minimum of motif length
    set repeat length (L) to previous[j-1];
    set first repeat end to i-1;
    set second repeat end to j-1;
 ENDIF
END ELSE
```

Advantage of using hash table: Since the current and previous vectors are sparse, the recursive operation at each i (along Y-axis) and j (along X-axis) takes more time for longer sequence. In order to optimize the execution time, a new methodology has been implemented for scanning phase using hash table. The above recursive operation is carried out for each i against X-axis and is only for some j's which are the positions of character (S[i]) in string S. i.e., Voccurences[S[i]] (see Fig. 1). It is explained by using the following lemma: Lemma 1 states: for each i (0$\leq$i$\leq$N), the algorithm checks only the positions next to the previous repeat (see Fig. 1) and at all positions of character S[i], instead for all j (0≤i≤N). As a proof, there can be only three possibilities at each i, such as: (A) any previous repeat can be continued or extended (extended repeat), (B) previous repeat can be terminated and are needed for output (terminated repeat) and finally (C) any new repeat can be a start (see Fig. 1). The above three repeat possibilities are further classified in two sections: (a) for possibility A and C, the match of S[i] and S[j] need to be identical. Further, the current vector of j^{th} element attains the length L > 0, represents the current repeat ($R_{i,j,L}$). Thus, the current repeat is a start position of a repeat or in the part of a continuous or extended repeat. (b) In case of B, termination of previous repeat ($R_{i-1,j-1,L}$), S[i] and S[j] does not match or the length of j equal to the string S. It means that the next position to the previous repeat does not match with the positions of S[i] in (Voccurence[S[i]]) or the repeat is terminated at the end of the sequence and are referred as terminated repeats ($R_{i-1,j-1,L}$). Thus, the



i \ j	A	E	R	M	S	G	T	F	A	E	R	E	R	I	L	S	M	S	G	A	E	R	M	S
A	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-
E	-	2	-	-	-	-	-	-	-	2	-	1	-	-	-	-	-	-	-	-	2	-	-	-
R	-	-	3	-	-	-	-	-	-	-	3	-	2	-	-	-	-	-	-	-	-	3	-	-
M	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	4	-
S	-	-	-	-	5	-	-	-	-	-	-	-	-	-	-	1	-	2	-	-	-	-	-	5
G	-	-	-	-	-	6	-	-	-	-	-	-	-	-	-	-	-	-	3	-	-	-	-	-

Fig. 1. A sample sequence is aligned along X- and Y- axes in a two-dimensional space. The number of repeat possible (current, previous and terminated) is highlighted.

algorithm scans only in the regions of positions next to previous repeat and all positions of character S[i]. The following steps are carried for each iteration of i with respect to lemma 1;

1. updation of current vector due to current repeat from (a) of lemma 1. current[j]=previous[j-1]+1; $\forall$ j $\in$ Voccurrence[S[i]]
2. finding terminating repeat from (b) of lemma 1. if(previous[j-1] > minimum length of repeat AND j $\notin$ Voccurrence[S[i]]) then Vterminated.push(j-1); $\forall$ j-1 $\in$ Voccurrence S[i-1]]

The Vterminated vector stores all the end positions of terminated repeats. Moreover, the algorithm performs the above two operations together by merging Voccurence[S[i]] and Voccurence[S[i-1]].

Pseudo code (entire searching phase using hash table)

```
WHILE (m < Voccurrence[S[i]].size() && n < Voccurrence[S[i-1]].size() )
 IF Voccurrence [S[i]][m]] == Voccurence [S[i-1]][n]]+1) THEN
 //current repeat
 set current[Voccurrence[S[i]][m]] to previous[Voccurrence [S[i-1][n]]+1;
 set m to m + 1; set n to n + 1;
 ENDIF
 ELSE IF Voccurence[S[i]][m]] < Voccurence[S[i-1]][n]+1] THEN
 //current repeat with no previous repeat found
 current[Voccurence[S[i][m]]=1;
 Set m to m+1;
 END ELSE IF
 ELSE IF Voccurence[S[i][m]] < Voccurence[S[i-1]][n]]+1 THEN
 // no current repeat found and this is terminating repeat
 IF previous[Voccurence[S[i-1][n]] >= minimum length of repeat THEN
 Vterminated.push(Voccurence[S[i-1][n]]);
 ENDIF
 Set n to n+1;
 END ELSEIF
 ENDWHILE
```

2.3 Post Processing Phase

In this section, for each terminated repeat ($R_{i-1,j-1,L}$) in Vterminated vector, the repeat length (L) is checked against the length of all the repeats in a previous vector. If the length is greater than or equal to L (terminated repeat length), then all such previous repeat ($R_{i-1,j'-1,L}$) positions are stored in a data structure motif. These motif are pushed into a vector Vmotif. Further, the value of the vector Vmotif is sorted (on the basis of repeat string) using a built-in STL (Standard Template Library) function. Finally, unique motifs are determined after removing all the redundant entries. The detailed output of the algorithm contains the length of the motifs and their start and end positions.

2.4 Time Complexity

The computational or time complexity of the algorithm is explained below based on the following; **Preprocessing:** In this phase, the positions of each alphabets in the string S is identified to create a hash table and the scanning process is performed in one-dimensional space with O(N) time complexity, where N is the length of the string S. **Scanning:** As explained earlier, the algorithm performs the scanning operations together by merging Voccurence[S[i]] and Voccurence[S[i-1]]. Thus for each i, the scanning sequence along X-axis takes $O(2N/|\sum|)$ time and $\sum$ represents the alphabets in string S, where $|\sum|$ represents the size of $\sum$ set. During each iteration along Y-axis, the value of current vector is assigned into previous vector and the current vector is reinitialized to zero, results in $O(N/|\sum|)$ time complexity. In addition, the process of Vterminated repeats requires scanning along X-axis results again in $O(N/|\sum|)$ complexity. Thus, the entire scanning phase takes $O(4N/|\sum|)$ time to find identical repeats in the string S. **Post processing:** In this section, the motif stored in Vmotif is sorted using the STL sort function which results in (N'logN') time complexity (where N' is number of repeats). However, the execution time of STL sort is less compared to that of the above steps. Considering the above three cases, the algorithm follows $O(4N^2/|\sum|)$ time complexity to find the identical repeats using hash table. The algorithm is more effective with an increase in $|\sum|$ and is improved over the existing algorithm, FAIR [28].

3 Results and Discussion

3.1 Case Study 1

To test the efficiency of the proposed algorithm, a set of eight major CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) nucleotide sequences is considered in this case study. The CRISPRs are direct repeats (identical repeats) with a length ranges from 24 to 48 nucleotides and the repeats in DNA are separated by spacers of similar length. These repeats are commonly present in many bacteria and archaea groups which help for the acquired resistance against phages [29]. The CRISPR sequences used in the present study are taken from four different species such as *Salmonella typhimurium* LT2*, Salmonella enteric serovar Typhi* Ty2*, Salmonella enteric serovar Paratyphi* A Str. AKU_12601 and *Salmonella enteric Choleraesuis.* The sequences are of various lengths with a minimum and maximum of 212 and 1982 nucleotides respectively. The input parameters provided for the search are: (a) the length of motif to be searched (for example: greater than or equal to 30) and (b) the minimum number of motif multiplicity (for example: greater than or equal to two). The motif multiplicity is defined as the number of times a motif is repeated in all the given sequences. The proposed algorithm identified 118 possible motifs in all four CRISPR sequences from four different species (*Salmonella typhimurium*LT2 , *Salmonella enteric serovar Typhi* Ty2, *Salmonella enteric serovar Paratyphi* A Str. AKU_12601 and *Salmonella enteric Choleraesuis*). A sample output (only a part of the output is shown for clarity) of the result is shown below.

```
----------------------------------------------------------------------
Input file name: fasta.txt
Length of motif: greater than 30
Motif multiplicity: greater than 2
Output file name: out.txt
----------------------------------------------------------------------
Motif: AACGGTTTATCCCCGCTGGCGCGGGGAACAC
Motif length: 31
Motif Occurrences :7
Present in 3 Sequences
>CRISPR-1, SALMONELLA TYPHIMURIUM LT2
Position(s): [304,334]
>CRISPR-2, SALMONELLA TYPHIMURIUM LT2
```

```
Position(s): [427,457] [549,579]
>CRISPR-3, SALMONELLA TYPHIMURIUM LT2
Position(s): [243,273] [793,823] [1586,1616] [1891,1921]
----------------------------------------------------------------------
Motif : CGGTTTATCCCCGCTGGCGCGGGGAACACA
Motif length :30
Motif Occurrences:15
Present in 6 Sequences
>CRISPR-1, SALMONELLA TYPHIMURIUM LT2
Position(s): [306,335]
>CRISPR-2, SALMONELLA TYPHIMURIUM LT2
Position(s): [673,702]
>CRISPR-3, SALMONELLA TYPHIMURIUM LT2
Position(s): [612,641] [856,885] [1039,1068] [1100,1129]
             [1405,1434] [1466,1495]
>CRISPR-2, SALMONELLA CHOLERAESUIS
Position(s): [306,335]
>CRISPR-1, SALMONELLA PARATYPHI A
Position(s): [184,213] [306,335] [367,396]
>CRISPR-1 SALMONELLA TYPHI TY2
Position(s): [62,91] [184,213] [245,274]
----------------------------------------------------------------------
Motif : GCGGTTTATCCCCGCTGGCGCGGGGAACAC
Motif length :30
Motif Occurrences :23
Present in 5 Sequences
>CRISPR-2, SALMONELLA TYPHIMURIUM LT2
Position(s): [123,152] [489,518] [611,640] [672,701] [733,762]
             [795,824] [856,885]
>CRISPR-3, SALMONELLA TYPHIMURIUM LT2
Position(s): [61,90] [183,212] [611,640] [733,762] [1038,1067]
             [1099,1128] [1221,1250] [1343,1372] [1709,1738]
>CRISPR-2, SALMONELLA CHOLERAESUIS
Position(s): [244,273]
>CRISPR-1, SALMONELLA PARATYPHI A
Position(s): [122,151] [183,212] [244,273] [427,456]
>CRISPR-1 SALMONELLA TYPHI TY2
Position(s): [122,151] [244,273]
----------------------------------------------------------------------
```

It is interesting to note that, the above motif of length 30 residues, GCG-GTTTATCCCCGCTGGCGCGGGGAACAC, clearly shows the efficiency of the proposed algorithm in finding the motif in all possible locations of the chosen four nucleotide sequences. Firstly, the different motif locations identified in the sequences of CRISPR-2 of *Salmonella typhimurium* LT2 and CRISPR-1 of *Salmnoella Paratyphi* A are found to be separated by an approximate spacer of length 32 nucleotides. Further, it is also to note that the occurrence of the motif is nearly conserved at the same locations (123 to 152) and (244 to 273). However, the number of motifs in each sequence varies (minimum = 1 and maximum 9) and represent genome variations among the four species.

3.2 Case Study 2

A total of three hexokinase-1 protein sequences from orthologous species such as *Homo Sapiens, Mus Musculus* and *Rattus norvegicus* are considered in this case study. The minimum length of motif to be searched is given as greater than or equal to 5 and the motif multiplicity is given as two (by default). The proposed algorithm identified 85 identical motifs (only part of the output is shown below) present in all the three sequences. Interestingly, a total of 17 out of 85 identified motifs are repeated more than once in the same protein sequence (see below for details). A sample output of the repeat motifs (17) is shown below.

```
----------------------------------------------------------------------
Input file name: fasta.txt
Length of motif: greater than 5
Motif multiplicity: greater than 2
Output file name: out.txt
----------------------------------------------------------------------
Motif : FVRSIPDG
Motif length :8
Motif Occurences:4
Present in 3 Sequences
>gi|188497754|REF|NP_000179.2|[HOMO SAPIENS]
Position(s) : [67,74]
>gi|148700161|GB|EDL32108.1| [MUS MUSCULUS]
Position(s) : [66,73]
>gi|6981022|REF|NP_036866.1| [RATTUS NORVEGICUS]
Position(s) : [67,74] [515,522]
----------------------------------------------------------------------
Motif : GSGKGAA
Motif length :7
Motif Occurrences:5
Present in 3 Sequences
>gi|188497754|REF|NP_000179.2|[HOMO SAPIENS]
Position(s) : [448,454] [896,902]
>gi|148700161|GB|EDL32108.1| [MUS MUSCULUS]
Position(s) : [447,453] [895,901]
>gi|6981022|REF|NP_036866.1| [RATTUS NORVEGICUS]
Position(s) : [896,902]
----------------------------------------------------------------------
Motif : GFTFSFPC
Motif length :8
Motif Occurrences :6
Present in 3 Sequences
>gi|188497754|REF|NP_000179.2|[HOMO SAPIENS]
Position(s) : [151,158] [599,606]
>gi|148700161|GB|EDL32108.1| [MUS MUSCULUS]
Position(s) : [150,157] [598,605]
```

```
>gi|6981022|REF|NP_036866.1| [RATTUS NORVEGICUS]
Position(s) : [151,158] [599,606]
----------------------------------------------------------------
Motif : VAVVNDTVGTMMTC
Motif length :14
Motif Occurrences :6
Present in 3 Sequences
>gi|188497754|REF|NP_000179.2|[HOMO SAPIENS]
Position(s) : [204,217] [652,665]
>gi|148700161|GB|EDL32108.1| [MUS MUSCULUS]
Position(s) : [203,216] [651,664]
>gi|6981022|REF|NP_036866.1| [RATTUS NORVEGICUS]
Position(s) : [204,217] [652,665]
----------------------------------------------------------------
```

The above results clearly show that the repeat motifs are conserved in all three sequences used. It is interesting to note, the three-dimensional structure of the last two motifs (GFTFSFPC and VAVVNDTVGTMMTC) repeated twice in *Homo Sapiens* and are superposed well with a root mean square deviation of $0.17Å$ and $0.27Å$ [16]. Further, the above results have been compared with the results of sequence alignment programs such as BLASTP [30] and CLUSTALW [31]. The output (results not shown) of these programs shows that the orthologous sequences exhibit high sequence similarity of more than 95%. Thus, the sequences are aligned end to end which leads to complexity in identifying the repeated motifs.

4 Implementation

The algorithm requires three inputs: a file of nucleotide or protein sequences in FASTA format, the length of the sequence motif to be searched and the number of motif multiplicity. The proposed algorithm generates a detailed output containing the location of motifs in each sequence. An option is also provided for the users to remove the redundant entries from the given input sequences. For example, only one sequence will be considered if two of the given or uploaded input sequences are having sequence identity of more than or equal to 90%. Due to less time complexity of proposed algorithm, there is no limitation in the number of motifs to be identified. The proposed algorithm has been written in C++ and successfully tested on a Linux box (Fedora core 9 and Red hat 9.0) and Solaris (10.0) environments. A standalone version of the proposed algorithm can be obtained upon request by sending an E-mail to the corresponding author Dr. K. Sekar (sekar@physics.iisc.ernet.in). In the future, we also plan to create an internet computing server for the proposed algorithm.

5 Conclusion

The algorithm finds the identical motifs in both nucleotide and proteins sequences. It has been developed with a broad view in mind to provide a comprehensive solution to the task of finding conserved as well as direct repeat motifs in a given multiple biological sequences. Further, the algorithm helps to analyze the differences in repeat numbers in various genomes and provides an insight to the horizontal gene transfer events during microbial evolution. One of the potential applications of this work is the comparative study of transposons in different sub species which provides a trace for the analysis of gene duplication.

Acknowledgements

The authors gratefully acknowledge the use of the Bioinformatics Centre (DIC) and the Interactive Graphics Based Molecular Modeling facility (IGBMM). These facilities are supported by the Department of Biotechnology, Government of India, India. One of the authors (KS) thank the Department of Information Technology (DIT) for funding in the form of a research grant.

References

1. D'Haeseleer, P.: What are DNA sequence motifs? Nat. Biotechnol. 24, 423–425 (2006)
2. Kumar, C., Kumar, N., Sarani, R., Balakrishnan, N., Sekar, K.: A Method to find Sequentially Separated Motifs in Biological Sequences (SSMBS). In: Chetty, M., Ngom, A., Ahmad, S. (eds.) PRIB 2008. LNCS (LNBI), vol. 5265, pp. 13–37. Springer, Heidelberg (2008)
3. Hulo, N., Sigrist, C.J., Le Saux, V., Langendijk-Genevaux, P.S., Bordoli, L., Gattiker, A., De Castro, E., Bucher, P., Bairoch, A.: Recent improvements to the PROSITE database. Nucl. Acids Res. 32, D134–D137 (2004)
4. Huang, J.Y., Brutlag, D.L.: The EMOTIF database. Nucl. Acids Res. 29, 202–204 (2001)
5. Zdobnov, E.M., Apweiler, R.: InterProScan–an integration platform for the signature-recognition methods in InterPro. Bioinformatics 17, 847–848 (2001)
6. Bailey, T.L., Elkan, C.: Fitting a mixture model by expectation maximization to discover motifs in biopolymers. Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology 2, 28–36 (1994)
7. Rigoutsos, I., Floratos, A.: Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm. Bioinformatics 14, 55–67 (1998)
8. Werner, T.: Model for prediction and recognition of eukaryotic promoters. Mamm. Genome 10, 168–175 (1999)
9. VanHelden, J., Andre, B., Collado-Vides, J.: Extracting Regulatory Sites from the Upstream Region of Yeast Genes by Computational Analysis of Oligonucleotide Frequencies. J. Mol. Biol. 281, 827–842 (1998)
10. Koonin, E.V., Mushegian, A.R., Galperin, M.Y., Walker, D.R.: Comparison of archeal and bacterial genomes: Computer analysis of protein sequence predicts novel function and suggests chimeric origins for the archaea. Mol. Microbiol. 25, 619–637 (1997)
11. Boby, T., Patch, A.M., Aves, S.J.: TRbase: a database relating tandem repeats to disease genes in the human genome. Bioinformatics 21, 811–816 (2005)
12. Mojica, F.J., Diez-Villasenor, C., Soria, E., Juez, G.: Biological significance of a family of regularly spaced repeats in the genomes of archaea, bacteria and mitochondria. Mol. Microbiol. 36, 244–246 (2000)
13. Van de Lagemaat, L.N., Gagnier, L., Medstrand, P., Mager, D.L.: Genomic deletions and precise removal of transposable elements mediated by short identical DNA segments in primates. Genome Res. 15, 1243–1249 (2005)

14. Wu, T.T., Miller, M.R., Perry, H.M., Kabat, E.A.: Long identical repeats in the mouse gamma 2b switch region and their implications for the mechanism of class switching. EMBO J. 3, 2033–2040 (1984)
15. Banerjee, N., Chidambarathanu, N., Sabarinathan, R., Michael, D., Vasuki Ranjani, C., Balakrishnan, N., Sekar, K.: An Algorithm to Find Similar Internal Sequence Repeats. Curr. Sci. 97, 1345–1349 (2009)
16. Sarani, R., Udayaprakash, N.A., Subashini, R., Mridula, P., Yamane, T., Sekar, K.: Large cryptic internal sequence repeats in protein structures from Homo sapiens. J. Biosciences 34, 103–112 (2009)
17. Sabarinathan, R., Basu, R., Sekar, K.: ProSTRIP: A method to find similar structural repeats in three-dimensional protein structures. Comput. Biol. Chem. 34, 126–130 (2010)
18. Heringa, J.: Detection of internal repeats: How common are they? Curr. Opin. Struct. Biol. 8, 338–345 (1998)
19. Djian, P.: Evolution of simple repeats in DNA and their relation to human diseases. Cell 94, 155–160 (1998)
20. Pons, T., Gomez, R., Chinea, G., Valencia, A.: Beta-propellers: associated functions and their role in human diseases. Curr. Med. Chem. 10, 505–524 (2003)
21. MOTIF SCAN, `http://myhits.isb-sib.ch/cgi-bin/motif_scan`
22. de Castro, E., Sigrist, C.J., Gattiker, A., Bulliard, V., Langendijk-Genevaux, P.S., Gasteiger, E., Bairoch, A., Hulo, N.: ScanProsite: detection of PROSITE signature matches and ProRule-associated functional and structural residues in proteins. Nucl. Acids Res. 34, W362–W365 (2006)
23. Schultz, J., Milpetz, F., Bork, P., Ponting, C.P.: SMART, a simple modular architecture research tool: identification of signaling domains. Proc. Natl. Acad. Sci. USA 95, 5857–5864 (1998)
24. MOTIF Search, `http://motif.genome.jp/`
25. Hughes, J.D., Estep, P.W., Tavazoie, S., Church, G.M.: Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. J. Mol. Biol. 296, 1205–1214 (2000)
26. Neduva, V., Linding, R., Su-Angrand, I., Stark, A., de Massi, F., Gibson, T.J., Lewis, J., Serrano, L., Russell, R.B.: Systematic discovery of new recognition peptides mediating protein interaction networks. PLoS Biol. 3, e405 (2005)
27. Favorov, A.V., Gelfand, M.S., Gerasimova, A.V., Ravcheev, D.A., Mironov, A.A., Makeev, V.J.: A Gibbs sampler for identification of symmetrically structured, spaced DNA motifs with improved estimation of the signal length. Bioinformatics 21, 2240–2245 (2005)
28. Banerjee, N., Chidambarathanu, N., Michael, D., Balakrishnan, N., Sekar, K.: An Algorithm to Find All Identical Internal Sequence Repeats. Curr. Sci. 95, 188–195 (2008)
29. Sorek, R., Kunin, V., Hugenholtz, P.: CRISPR - a widespread system that provides acquired resistance against phages in bacteria and archaea. Nat. Rev. Microbiol., 181–186 (2008)
30. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Biol. 215, 403–410 (1990)
31. Thompson, J.D., Higgins, D.G., Gibson, T.J.: CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. Nucl. Acids Res. 22, 4673–4680 (1994)