MULTIDIMENSIONAL LONGEST COMMON SUBSEQUENCE DISCOVERY From LARGE DATABASE USING DNA OPERATIONS

B.LAVANYA [#1], A. MURUGAN [*2]
[#] Department of Computer Science, University of Madras
Chennai, India.
[1] lavanmu@gmail.com
[*] Department of Computer Science, Dr. Ambedkar Govt. College
Chennai, India
[2] amurugan1972@gmail.com

Abstract - The problem of analysis of biological sequences, is the discovery of sequence similarity of various kinds, in the primary structure of related proteins and genes. This sequence search can be applied to various applications like discovery of association rules, strong rules, correlations, sequential rules, frequent episodes, multidimensional patterns and many other important discovery tasks. In this paper we have proposed two new approaches to find multidimensional LCS and SCS, of N sequences parallely, using DNA operations. These approaches can be used to find MLCS and MSCS, of any window size, from any number of sequences, and from any type of input data. The proposed work can be applied to finding diverging patterns, constraint MLCS, and many more related patterns Implementation results shown the correctness of the algorithms. Finally, the validity of the algorithms are checked and their time complexity is analyzed.

Keywords: sequential patterns, DNA operations, Parallel algorithms.

I. INTRODUCTION

Analysis of biological sequences is the discovery of sequence similarity of various kinds, in the primary structure of related proteins or genes. The frequent subsequences usually correspond to residues conserved during the evolution due to important structural or functional behavior. This is the prime motivation for the proposed work. The task of discovering frequent subsequences as patterns in a sequence data base is done in [21,22,28,31,40]. Several methods have been proposed for dealing with sequence mining [5,6,9,16,26,35] A detail survey of several multiple-string alignment algorithms can be found in [11]. They encountered many notable problems like, the task of optimally aligning a set of strings is computationally very expensive[19] and they could align the global similarities[28]. If the sequences under comparison are distantly related or if the relative order of their similar regions varies among sequences, it is quite possible that no substantial alignment can be produced. To overcome the difficulty of alignment problem a modified Position Weight Matrices (PWM) [3] can be used to focus on the positions of the patterns in the sequences. Various ways of building a PWM have been carried out, some of them are found in [34,8,17,15,31]. Building modified PWM is given in [4].

II. LITERATURE REVIEW

A number of pattern discovery algorithms have been steadily appearing in the literature [1,4,6, 7,12,14,15,19,25,27]. Simulation of all the DNA operations are done in [2], the proposed work uses the DNA operations *cut* and *pcr* operations. Using DNA operations and modified PWM, given a sequential database different related patterns were discovered in [1,2,3,4,5,6]. In DNA sequence mining, Zhang et al [28] introduce gap requirement, in mining periodic patterns from sequences. In particular, all the occurrences (both overlapping ones and non overlapping ones) of a pattern in a sequence satisfying the gap requirement and different other patterns are captured, and the support is the total number of such occurrences are found in [3, 4]. This paper deals with finding MLCS of any window size, with given constraint, diverging pattern and contrast pattern [23,37,38]. This work is an extension of [6] where LCS is discovered using two new approaches. Multidimensional sequence mining is done in [13]**.** The authors have performed data parallelism to mine multidimensional sequence mining. The proposed work which is an extension of [6] performs both task and data parallelism, to discover MLCS and other related patterns.

A. *Definitions*

Definition1. (Subsequence and Landmark): Sequence S = {e_1, e_2, ...e_m } is a subsequence of another sequence S′ = {e'_1, e'_2, ...e'_n } ($m \le n$), denoted by $S \subseteq S'$ (or S′ is a super sequence of S) $1 \le l_1 \le l_2 \le ... \le l_m \le n$ such that S[i] = S′[l_i] (i.e., $e_i = e'l_i$) for i = 1, 2, ..., m. Such a sequence of integers $\langle l_1, l_2, ...l_m \rangle$ is called a landmark of S in S′.

A pattern P = e_1, e_2, ...e_m is also a sequence. For two patterns P and P ′, if P is a subsequence of P ′, then P is said to be a sub-pattern of P ′, and P ′ a super-pattern of P .

Definition 2. Instances of Pattern: For a pattern P in a sequence database SeqDB = S_1, S_2, ..., S_n, if $\langle l_1, e_2, ...l_m \rangle$ is a landmark of pattern P = e_1, e_2, ...e_m in Si ∈ SeqDB, pair (i, $\langle l_1, e_2, ...l_m \rangle$) is said to be an instance of P in SeqDB, and in particular, an instance of P in sequence Si.

Definition 3. Repetitive Support and Support Set: The repetitive support of a pattern P in SeqDB is defined to be sup(P) = max (I) where I Ɛ SeqDB(P) is non-redundant. The non-redundant instance set I with I = sup(P) is called a support set of P in SeqDB.

Definition 4. Position Weight Matrix: Given a finite alphabet Σ and a positive integer m, a PWM M is a matrix with ||Σ|| rows and m columns. The coefficient M, (p, x) gives the score at position p for the letter x in Σ. The PWM defines a function from σm to ʀ, that associates a score to each word u = {u1,u2,...,up} of σm :

ScoreM (u)=Σmp−1 M (p, up),

Let α be a score threshold. We say that M has an occurrence in a text T at position k if ScoreM (T_k ...T_{k+m-1}) ≥ α.

The most recurrent task is to predict binding sites in a large DNA sequence, that is to look for occurrences of a PWM, given a text.

Definition 5. Longest Common Subsequence: Given two sequences X = < x_1 , x_2, …, x_m > and Z =< z_1, z_2, …, z_k> , we say that Z is a subsequence of X, if there is a strictly increasing sequence of k indices < $i_{1,}$ $i_{2,}$ ..., i_k > ($1 \le i_1 \le i_2 \le ... \le i_k \le n$) such that Z = < x_1, x_2, … x_{ik} >.

For example, let X = < ABRACADABRA > and let Z =< AADAA >, then Z is a subsequence of X. Given two strings X and Y for example, let X be as before and let Y = <YABBADABBADOO>. Then the LCS is Z = < ABADABA >, refer Figure 1. There are many solutions for finding LCS like dynamic programming solution [37], Hunt-Szymanski algorithm [23], etc.

S1	**H U M A N**
S2	**C H I M P A N Z E E**
LCS	**H M A N**

Fig. 1. Example of Longest Common Subsequence

Definition 6. Dimension: A table T is a set of tuples <D,A_1,A_2,A_3,…,A_N>, where D is an attribute whose domain is totally ordered. A sequence S is denoted by an ordered list <t_1,t_2,t_3,…,t_k>, where t1 is a tuple, i.e D(t_1)≤ D(t_2)≤ D(t_3)≤…. D(t_k) for 1 ≤i ≤k and D(t_1) = value of D tuple t_1. Every tuple has n analysis attributes along with an ordered value. A set of analysis attribute values can occur at most once in a same value of D, but can occur multiple times in different values of D attribute. The number of values of D in a sequence is called the length of the sequence. If the length of S in k, then we call it a k-sequence [13].

A multidimensional sequence database is of schema<D, A_1,A_2,A_3,…,$A_{n,}$ R_1,…,R_m>, where R_i are called relevant dimensions. The schema is partitioned according to relevant attribute values and support computed by number of partitions that contain sequence. As mentioned previously, every partition is similar to table T, a set of tuples <D, A_1,A_2,A_3,…,A_n > [13]. Given a minimum support threshold min-support, a multidimensional sequence S is called a multidimensional sequential pattern if and only if support (S) ≥ min-support.

III. DNA OPERATIONS BASED ALGORITHMS

In this paper, we propose, two new approaches to study the Multidimensional Longest Common Subsequences [MLCS] mining problem and other different related patterns. Algorithms 1 and 2 searches for all common sequences of different window sizes and different other patterns, in input sequences, using support vector and modified Position Weight Matrices (PWM).

Our approaches makes minimal assumptions about the background sequence model and the mechanism by which elements affect gene expression. This provides a versatile motif discovery method, across all data types and genomes, with exceptional sensitivity and near-zero false-positive rates. Our algorithms does not use any complex statistical models but rather uses DNA operations and DNA strands to search for the presence or absence of patterns. The exponential nature of some PWM problems, is a limiting factor for using matrices of medium or large length. Here, we use DNA strands to store large data and DNA operations to access them parallely [2][3], thus solving the above noted problem.

A. Finding MLCS using Support Vector (MLCSSV)

Algorithm MLCSSV discovers MLCS and different related patterns, with its support vector using DNA operations. Algorithm 1with window size 3 is illustrated in Figure 2.

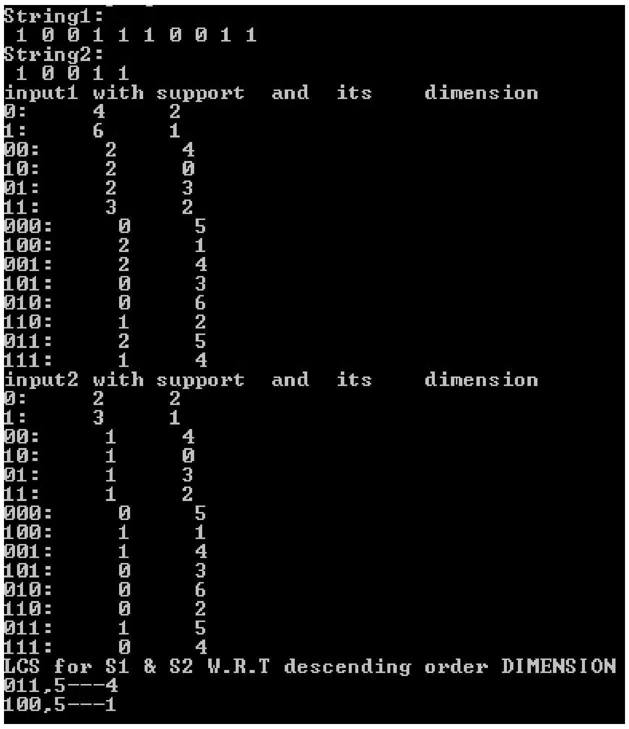
Algorithm 1: DNA-based-MLCS discovery using Support Vector (MLCSSV).

Input: S, level_number, D

Output: MLCS strand, supp strand

```
1.  begin
2.    Generate $DNA_0$ and $DNA_1$;
3.    number_of_ nodes ← size($DNA_0$) ;
4.    $DNA_{01}...DNA_{0N}$ ← pcr($DNA_0$) ;
5.    $DNA_{11}...DNA_{1N}$ ← pcr($DNA_1$) ;
6.   foreach element of $DNA_{01}...DNA_{0N}$ and $DNA_{11}...DNA_{1N}$ do
7.     Create threads parallely ;
8.     let $supp_{01}...supp_{0N}$[],$pos_{01}...pos_{0N}$[][]← cut(S,$DNA_{01}...DNA_{0N}$[element]) ;
9.     let $supp_{11}...supp_{1N}$[],$pos_{11}...pos_{1N}$[][]← cut(S,$DNA_{11}...DNA_{1N}$[element]) ;
10.  end
11.  [parallely for lcs0 and lcs1] ;
12.  foreach j from 1 to number_ of_ nodes do
13.     if ($supp_{01}$[j]…$supp_{0N}$[j]) > 0 then
14.         lcs0[] ← $DNA_{01}$[j];
15.         supp1[] ← min($supp_{01}...supp_{0N}$);
16.     end
17.     if ($supp_{11}$[j]…$supp_{1N}$[j]) > 0 then
18.        lcs1[] ← $DNA_{11}$[j] ;
19.        supp2[]← min($supp_{11}...supp_{1N}$);
20.     end
21.  end
22.  foreach (lcs0[] or lcs1[]) <> 0
23.    if ((lcs0[]. $S_1.D_1$[]) = = (lcs0[]. $S_2.D_2$[])) then
24.        MLCS[] ← lcs0[];
25.    if ((lcs1[].$S_1.D_1$[]) = = (lcs1[]. $S_2.D_2$[])) then
26.        MLCS[] ← lcs1[];
27.  end
28.    Extended for any number of Sequences;
29.  end
```

Algorithm MLCSSV discovers MLCS and different related patterns, with its support vector using DNA operations, refer Figure 3. Let S = (s_1,s_2...s_N), be the N input sequences, encoded in 0's and 1's, D = (D_1,D_2...D_N) be their dimensions respectively and the level_number be the maximum length of MLCS sequence required, that is, the window size of MLCS. MLCS strand along with its supp, that is, number of times each subsequence is present in the given sequences be the output strands (support). Step 2 generate DNA_0, the possible combinations of 0's and DNA_1, the possible combinations of 1's [18], depending on the level_number. Let the number_of_nodes be the variable, which stores the total number of elements present in DNA_0 or DNA_1. Steps 4 and 5 performs the pcr operation on DNA_0 and DNA_1 strands, for N sequences and stores in DNA_$_{01}$...DNA_{0N} and DNA_{11}...DNA_{1N} respectively. In steps 6 to 10, for each element of DNA_{01}...DNA_{0N} and DNA_{11}...DNA_{1N}, threads are created parallely and cut operation is applied to find the support count and position of its occurrences and stored in $supp_{01}$...$supp_{0N}$, pos_{01}...pos_{0N}, $supp_{11}$...$supp_{1N}$ and pos_{11}...pos_{1N} respectively. In steps 12 to 21, the supp and pos strands are searched vertically for occurrences of all common subsequences for all window sizes and the LCS is found for the given level_number . Finally the discovered LCS in S are relationally checked for the equality of their respective dimensions and stored in MLCS strand. The same can be checked for ascending and descending order dimensions also. The Algorithm 1 depicts the process for two sequences in S, and can be extended for N sequences.



```
String1:
 1 0 0 1 1 1 0 0 1 1
String2:
 1 0 0 1 1
input1 with support  and  its    dimension
0:     4     2
1:     6     1
00:     2     4
10:     2     0
01:     2     3
11:     3     2
000:     0     5
100:     2     1
001:     2     4
101:     0     3
010:     0     6
110:     1     2
011:     2     5
111:     1     4
input2 with support  and  its    dimension
0:     2     2
1:     3     1
00:     1     4
10:     1     0
01:     1     3
11:     1     2
000:     0     5
100:     1     1
001:     1     4
101:     0     3
010:     0     6
110:     0     2
011:     1     5
111:     0     4
LCS for S1 & S2 W.R.T descending order DIMENSION
011,5---4
100,5---1
```

Fig. 2. Illustration of Algorithm 1

Time Complexity

The time complexity of Algorithm1 can be analyzed in 2 phases. The phase 1 is discovery of LCS [6] and discovery of MLCS from LCS constitutes phase 2. Therefore,

TC(MLCSSV) = O(LCS) + O(MLCS).

If levelnumber $\neq$ 0, then

TC(LCS) is between O((n/L) + n) and O(levelnumber)) at its best case

And between (O(n/M)+O((n/L)+n)) and O(levelnumber) at average case [6]

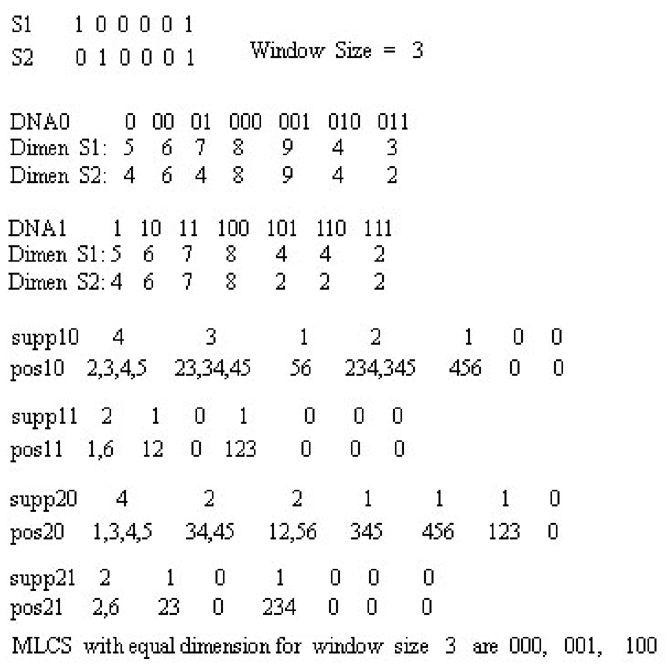
TC(MLCS) = O(|LCS|)

Therefore at its best case

TC(MLCSSV) = max((O((n/L) + n) and O(levelnumber)), O(|LCS|))

At its average and worst case

TC(MLCSSV) = max((O(n/M)+O((n/L)+n)) and O(levelnumber)), O(|LCS|)).



```
S1     1 0 0 0 0 1
S2     0 1 0 0 0 1        Window Size =  3

DNA0        0  00  01  000  001  010  011
Dimen S1:  5   6   7   8    9    4    3
Dimen S2:  4   6   4   8    9    4    2

DNA1       1  10  11  100  101  110  111
Dimen S1: 5   6   7   8    4    4    2
Dimen S2: 4   6   7   8    2    2    2

supp10    4         3         1        2         1    0    0
pos10   2,3,4,5   23,34,45    56    234,345    456    0    0

supp11   2     1    0    1     0    0    0
pos11   1,6    12   0   123    0    0    0

supp20    4        2        2       1       1       1     0
pos20   1,3,4,5   34,45   12,56    345     456     123    0

supp21   2      1     0     1     0    0     0
pos21   2,6     23    0    234    0    0     0
MLCS  with equal dimension for  window  size  3  are 000,  001,   100
```

Fig. 3. Finding MLCS using Support Vector (MLCSSV)

Special Case: Multidimensional Shortest Common Subsequence (MSCS): Algorithm MLCSSV can be used to find MSCS in S. Since all possible common sequences of all window sizes from 1 to level_number is generated, the Algorithm MLCSSV can be used to find common sequence of any small length, thus SCS and thereby MSCS. MSCS is found by varying the window size, that is 1.

B. *Finding MLCS using modified PWM (MLCSPWM)*

DNA based MLCS discovery using modified PWM, discovers MLCS in the given N sequences using DNA operations and modified PWM, refer Figure 4.

Algorithm 2: DNA-based-MLCS discovery using modified PWM (MLCSPWM)

Input: S, D

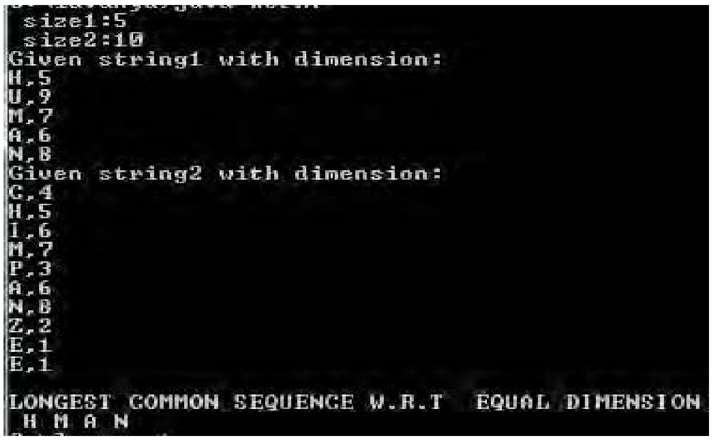
Output: MLCS strand, PWM strand

```
1.   begin
2.     L ← min(S);
3.     $T_1, T_2, …, T_N$ ← pcr(S);
4.     $PWM_1$[1…L]← cut($T_1$ , $s_L$[element]);
5.     $PWM_2$[1…L]← cut($T_2$ , $s_L$[element]);
6.     … $PWM_N$[1..L] ← cut($T_N$ , $s_L$[element]);
7.     j ←1;
8.     foreach i ranging from 1 to L do
9.        if (PWM1[i][j] > 0) then
10.            test ← $PWM_2$[i][0];
11.            lcs[][] ← i;
12.            lcs[][] ← $PWM_1$[i][j];
13.      end
14.     foreach (i ranging from i + 1 to L) AND ($PWM_{1..N}$ [i][j] ≠ ø) do
15.          if (test < $PWM_2$[i][j]) then
16.               test← $PWM_2$[i][j];
17.               lcs[][0] ← i;
18.               lcs[][1] ←$PWM_2$[i][j] ;
19.            end
20.            else
21.                   j++ ;
22.             end
23.       end
24.    foreach (lcs[][0] <> 0)
```

```
25.        if  ((lcs[][0]. $S_1$.$D_1$[]) = = (lcs[][0]. $S_2$.$D_2$[])) then
26.                 MLCS[] ← lcs[][0];
27.      end
28.             Extended for N number of PWM strands
29.     end
30.   end
```

Let S = ($s_{1,}$ s_2,…s_N), be the N input sequences, D = (D_1,D_2...D_N) be their dimensions respectively, and MLCS and PWM are the output strands, refer Figure 5. Step 2 finds the length of the smallest sequence in S and stores in L. Step 3 performs the pcr operation on each of the sequence in S and stored in $T_{1,}$ T_2,…,T_N. Steps 4-6 does the cut operation on $T_{1,}$ T_2,…,T_N, for each of s_L[element] and their position weight matrices are generated as PWM_1[1…L], PWM_2[1…L], ... PWM_N[1…L] respectively. Steps 8 to 25 performs a vertical check operation on all PWM, checking for the occurrences of elements of s_L, in order of their presence, stores in LCS strand, finally the discovered LCS elements in each sequence in checked for their dimensions with other sequences and stored in MLCS strand. Algorithm MLCSPWM illustrates for PWM1 and PWM2 strands, thus can be extended for N number of PWM strands .

Algorithm MLCSSV and MLCSPWM can be used to generate MLCS for different support counts, for any window size, and all MLCS with position of its occurrences, discover Multidimensional Constrained Longest Common Subsequence (MCLCS) and find diverging and emerging patterns with given dimensions.



```
 size1:5
 size2:10
Given string1 with dimension:
H,5
U,9
M,7
A,6
N,8
Given string2 with dimension:
G,4
H,5
I,6
M,7
P,3
A,6
N,8
Z,2
E,1
E,1

LONGEST COMMON SEQUENCE W.R.T  EQUAL DIMENSION
 H M A N
```

Fig. 4. Illustration of Algorithm 2

Time Complexity

The time complexity of Algorithm 2 can be analyzed in 2 phases. The phase 1 is discovery of LCS [6] and discovery of MLCS from LCS discovered constitutes phase 2. Therefore,

TC(MLCPWM) = O(LCS) + O(MLCS).

If PWM $\not\subset$ ø , then

TC(LCS) is between O((n/L)+n) and O(|min(S)|) at its best case,

and is between (O(n/M) + O((n/L) + n)) and O(|min(S)|) at its average case [6]

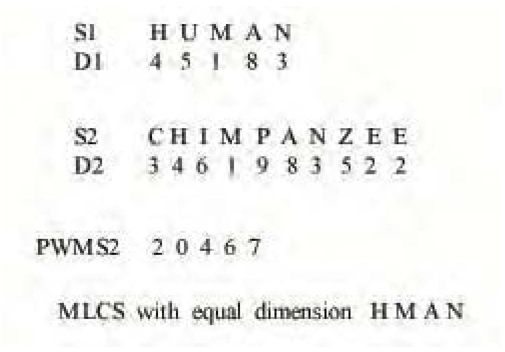
TC(MLCS) = O(|LCS|)

Therefore at its best case

The TC(MLCSPWM) = max(O((n/L)+n) and O(|min(S)|), O(|LCS|))

At its average and worst case

The TC(MLCSPWM) =max(O(n/M) + O((n/L) + n) and O(|min(S)|), O(|LCS|))



```
S1    H U M A N
D1    4 5 1 8 3

S2    C H I M P A N Z E E
D2    3 4 6 1 9 8 3 5 2 2

PWMS2   2 0 4 6 7

MLCS with equal dimension  H M A N
```

Fig. 5: DNA-based-MLCS discovery using modified PWM (MLCSPWM)

IV. DIFFERENT RELATED PATTERNS

Special Case 1: Find MCLCS and Sequence Divergence

Algorithms 1 and 2 can be extended to find MCLCS for given S. Since all possible LCS are found, the constraint can be can be applied and the final MCLCS can be found as shown in Figure 6 and thereby find sequence divergence also.

S1 **T A G T C A C G**

D1 **5 5 5 5 5 5 5 5**

S2 **A G A C T G T C**

D2 **5 5 5 5 5 5 5 5**

C = A T

Possible LCS of window size 4 is A G A C & A G T C

MCLCS = A G T C

Fig. 6. Example of MLCS

Special Case 2: Find Diverging and Emerging Patterns

Algorithms MLCSSV and MLCSPWM can also be used to find diverging and emerging patterns for given S. Steps 13 to 20 in Algorithm MLCSSV and steps 14 to 23 in Algorithm MLCSPWM, can be modified to find diverging and emerging patterns for given S.

Special Case 3: Re description Mining

The goal of re description mining is to use the given descriptors as a vocabulary and find subsets of data. Algorithms 1 and 2 can be extended to find subsets from a given set of data.

V. PERFORMANCE

Algorithms MLCSSV and MLCSPWM have been implemented and tested with simulated and real databases. Therandom DNA sequences of size varying from 100 to 25000, are generated from http://old.dnalc.org/bioinformatics/dnalc-nulceotide-analyzer.htm#randomizerand http://old.dnalc.org/bioinformatics.org/sms/rand-dna.html. The real data is collected from EMBL database in FASTA format. The genome sequences of 3021 viruses are collected and tested for the existence of all required patterns. The database is got from http://www.ebi.ac.uk/genomes/virus.html. Algorithms MLCSSV and MLCSPWM proved to be efficient and accurate in solving MLCS and MCLCS in the given sequences. Tested with randomly generated and real motifs, our work could discover all motifs present, with its positions of existence. All implementations are performed on a dual core computer and 5 GB main memory using Java. The operating system is Windows XP. The resulted data of these experiments are consistent. The limitation of these algorithm is that the maximum number of threads generated, is dependent on the efficiency of the system architecture.

VI. APPLICATIONS

The assumption behind the discovery of patterns is that a pattern that appears often enough in a set of biological sequences is expected to play a role in defining the respective sequences functional behavior and evolutionary relationships. Since the proposed new algorithms use DNA strands for its DNA operations and other processing, the storage and retrieval processes can be implemented easily and parallely, whatever may be the size of the database. Since the applications for finding, the existence of subsequences given a large database of commercial or genetic information are unlimited, the searching for MLCS and MCLCS has its importance in many industrial, research and scientific applications. Especially in medical and genetic field, the finding of all patterns of motifs with its diverging pattern, can be used to predict, analysis, interpret and conclude the existence or future liability of any disease or abnormality present in the patient data or defaulters in any commercial databases. This work can also be applied to analysis of rule based systems, expert systems, rule mining, pattern mining, program execution traces, algorithm behavioral patterns and other commercial database analysis.

VII. CONCLUSION

In this paper, we have designed and performed the implementations to find MLCS, MSCS, and different related patterns, in a highly parallel way, and can be extended to many other data mining applications also. In future, it is possible to solve more real time problems in molecular biology.

REFERENCES

[1] Murugan. A. and Lavanya. B. DNA algorithmic approach to solve GCS problem. *Journal of Computational Intelligence in Bioinformatics*,3(2):239-247, 2010.

[2] Murugan. A., Lavanya. B., and Shyamala. K. A novel programming approach for DNA computing. *International Journal of Computational Intelligence Research*, 7(2):199-209, 2011.

[3] Lavanya. B. and Murugan. A. Discovering sequence motifs of different patterns parallelly using DNA operations. *International Journal of Computer Applications*, 3(1):18-24, Nov 2011.

[4] Lavanya. B. and Murugan. A. A DNA based approach to find closed repetitive gapped subsequence from a sequence database. *International Journal of Computer Applications,* 29(5):45-49, Sep 2011.

[5] Needleman. S. B. and Wunsch. C. D. A general method applicable to the search of similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443-453, 1970.

[6] Lavanya. B. and Murugan. A. Mining Longest Common Subsequence and other related patterns using DNA Operations, *International Journal of Computer Applications,* 49(18): 38-44,2012.

[7] Nikhil Bansal, Moshe Lewenstein, Bin Ma, and Kaishong Zhang. On the longest common rigid subsequence problem. *Algorithmica,* 56:270-280, 2010.

[8] Maier. D. The complexity of some problems on subsequences and super sequences. *ACM*, 25:322-336, 1978.

[9] Wu. T. D. and Brutlag. D. L. Identification of protein motifs using conserved amino acid properties and partitioning techniques. Proceedings of the 3rd International conference on *Intelligent Systems for Molecular Biology*, pages 402-410, 1995.

[10] Isabelle da Piedade, Man-Hung Eric Tang, and Olivier Elemento. DISPARE: discriminative pattern refinement for position weight matrices. *BMC Bioinformatics*, 10(388):1471-2105, 2009.

[11] Hirosawa et al. Comprehensive study on iterative algorithms of multiple sequence alignment. *Computational Applications in Biosciences*, 11:13-18, 1995.

[12] Neuwald. A. F. and Green. P. Detecting patterns in protein sequences. *Journal of Molecular Biology*, 239:698-712, 1994.

[13] Mahdi Esmaieli and Mansour Tarafdar, Sequential Pattern Mining from multidimensional sequence data in parallel, *International journal of Computer theory and engineering*,2:5,730-733,2010.

[14] Smith. T. F. and Waterman. M. S. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195-197, 1981.

[15] Benson. G. and Waterman .M.S. A method for fast database search for all k-nucleotide repeats. 2nd International conference on *Intelligent Systems for Molecular Biology*, pages 83-98, 1994.

[16] Neville-Manning. C. G., Sethi. K .S., Wu. D.,and Brutlag. A. D. Enumerating and ranking discrete motifs. Proceedings of *Intelligent Systems for Molecular Biology*, pages 202-209,1997.

[17] Stormo. G. DNA binding sites: representation and discovery. *Bioinformatics*, 16:16-23, 2000.

[18] Kyle Jensen. L., Mark Styczynski. P., Isidore Rigoutsos, and Gregory Stephanopoulos. V. A generic motif discovery algorithm for sequential data. *Bioinformatics*, 22(1):21-28, 2006.

[19] Wang. L. and Jiang. T. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1:337-348, 1994.

[20] Nan Li and Tompa. M. Analysis of computational tools for motif discovery. *Algorithms of molecular biology*, pages 1-8, 2006.

[21] Lo. D.and Khoo. S. D. Liu. Efficient mining of iterative patterns for software specification discovery. Int. Conf. on *Knowledge Discovery and Data Mining*, pages 460-469, 2007.

[22] Annila. H. M., Toivonen. H., and Verkamo. A.I. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259-289, 1997.

[23] Landau. G. M., Levy. A., and Newman. I. LCS approximation via embedding into locally nonrepetitive strings. I*nformation and Computation*, 209:705-716, 2011.

[24] Li. M., Ma. B., and Wang. L. On the closest string and substring problems. *J. ACM*, 49(2):151-171, 2002.

[25] Martinez. M. An efficient method to find repeats in molecular sequences. *Nucleic Acid Research*, 11:4629-4634, 1983.

[26] Martinez. M. A flexible multiple sequence alignment program. *Nucleic Acid Research*, 16:1683-1691, 1988.

[27] Suyama. M., Nishioka. T., and Junichi. O. Searching for common sequence patterns among distantly related proteins. *Protein Engineering*, 8:1075-1080, 1995.

[28] Zhang. M., Kao. B., Cheung. B., and Yip. K. Mining periodic patterns with gap requirement from sequences. SIGMOD Int. Conf. on *Management of Data*, pages 623-633, 2005.

[29] Bin Ma. A polynomial time approximation scheme for the closest substring problem. *LCNS Springer*, 1848:99-107, 2000.

[30] Smith. H. O., Annau. T. M., and Chandrasegaran. S. Finding sequence motifs in groups of functionally related proteins. Proceedings of *National Academy (USA)*, 87:826-830, 1990.

[31] Agarwal. R. and Srikant. R. Mining sequential patterns. Int.Conf. on *Data Engineering*, 1995.

[32] Agarwal. R. and Srikant. R. Mining sequential patterns: Generalizations and performance improvements. *Extending Data Base Technology*, pages 3-17, 1996.

[33] Staden. R. Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Research*, 12:505-519, 1984.

[34] Isisdore Rigoutsos and Aris Floratos. Combinatorial pattern discovery in biological sequences: the teiresias algorithm. *Bioinformatics,* 14(1):55-67, 1998.

[35] Waterman. M. S., Galas. D. J., and Arratia. R. Pattern recognition in several sequences: consensus and alignment. *Bulletin of Mathematical Biology*, 46:515-527, 1984.

[36] Saurabh Sinha. On counting position weight matrix matches in a sequence, with application to discriminative motif finding. *Bioinformatics*, 22(14):454-463, 2006.

[37] Yin-Te Tsai. The constrained longest common subsequence problem. *Information processing letter*, 88:173-176, 2003.

[38] Qian Wan and Aijun An. Diverging patterns: Discovering significant dissimilarities in large databases. *Technical Report* CSE-2008-10, Dec 2008.

[39] Guan. X. and Uberbacher. E. C. A fast lookup algorithm for detecting repetitive DNA sequences. Proceedings of the *pacific symposium on Bio computing*, pages 718-719, 1996.

[40] Yan. X., Han. J., and Afhar. R. Colspan: Mining closed sequential patterns in large datasets. SIAM Int. Conf. *Data Mining*, pages 166-177, 2003.