Chapman-Kolmogorov Relation Based Median String Algorithm for DNA Consensus Classification

Mohammad Shibli Kaysar
Deptartment of Computer Science and Engineering
Chittagong University of Engineering and Technology
Chittagong, Bangladesh
*shibli.kaysar@gmail.com

Mohammad Ibrahim Khan
Deptartment of Computer Science and Engineering
Chittagong University of Engineering and Technology
Chittagong, Bangladesh
muhammad_ikhancuet@yahoo.com

***Abstract*—Consensus string is the most frequent common pattern in a set of string. Consensus string is an important feature of DNA sequence. Many algorithm have been introduced to discover consensus string. Among them, median string algorithm is the most popular one. Basically, that is a brute force algorithm.DNA sequence is composed of a series of four letter alphabet ∑={a,c,g,t}. If the size of the consensus string is l, then the algorithm generates all the 4^l number of l length strings called motifs or l-mer. Then try to fit the motifs one by one with the sequence. In this paper we have discovered a way to reduce the search space using chapman kolmogorov relation. We found that, the proposed system can find the same consensus string within a shorter period of time than the time taken by the median string algorithm. As the l-mer size increases, the proposed system takes much less time than the median string algorithm. For l-mer size 7, we found the proposed system is 47 times faster than the median string algorithm.**

Keywords—DNA,consensus, motif,chapman-kolmogorov

I. INTRODUCTION

Gene regulatory binding motifs are short DNA sequences that control gene expression. The length of these motifs, are 6-15 base pare long. These motifs do not contain any specific starting and end point. Gene regulatory binding motifs can start from anywhere within the DNA sequence. Since the starting point is unknown, the end point also is unknown. The only difference between regulatory motif and random sequence of an equivalent length is that, regulatory motifs occur more frequently than random sequence. The location of regulatory motif may vary from sample to sample. Frequency of these motifs, also vary from sequence to sequence. To discover such regulatory motif from a DNA sequence involves finding such a continuation of pattern [1]. This can be illustrated in Fig. 1.

Any DNA sequence is a sequence of four symbol like a, c, g, and t. Searching for a particular motif in a DNA sequence involves searching a short pattern over the alphabet ∑={a,c,g,t}. The main idea of motif search can be divided into two parts a) exploitation of appropriate model for representation of motif sequence b) formulation of algorithm for motif search. The most widely used motif models are position weight metrics[2] and consensus sequence[3].Position weight metric typically use statistical method, report in a short time but there is no guarantee about a global optimum[4,5,6]. Exact algorithms use consensus model.

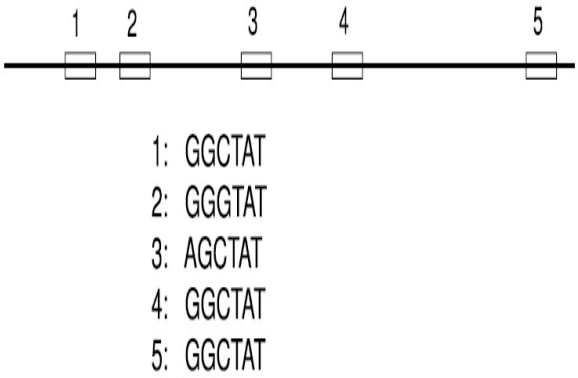

Fig. 1. Consensus string.

Most of the exact algorithms takes all string patterns of length l over alphabet ∑ as candidate motifs, and output the common pattern in all input sequences. In case exact algorithms, initially the search space is $O(|\Sigma|^l)$, that grows dramatically with the rise of $|\Sigma|$. As a result, most exact algorithms are developed for sequences where |∑|=4. But in case of protein data set where |∑|=20, these algorithms cannot find a low conserved motif within a reasonable time.

The idea of motif stem increased the potency of exact algorithms over large alphabet[7]. Consensus string (or closest string or center string) issues are to search out a representative string of a given set S of strings. The consensus issues are major issues in multiple string comparison and have been researched extensively [8,9,10,11,12,13,14,15] to unravel several issues arising in computational biology like motif finding, PCR primer style, and genetic probe style. Since most of the consensus issues are NP-complete, researchers even have designed fixed-parameter algorithms [10,16,17,18] approximation algorithms [13,17,19,20,21,22,23] and algorithms for a short range of strings [10,24,25].

II. MEDIAN STRING ALGORITHM FOR MOTIF SEARCH

Median string algorithm takes as input a set of t DNA sequences. It then generates all possible *l-mers* of length l. Since there are only 4 types of nucleotides in a DNA sequence, the number of all possible *l-mers* is 4^l. Then it starts from the start of each sequence, goes to the end by placing each *l-mer* and calculate distance. In this way for each *l-mer* ,

the minimum distance between the *l-mer* and each row is calculated. The sum of minimum distance of all the rows for a particular *l-mer* is the minimum distance of the *l-mer* with the DNA sample. In this way, the minimum distance for all the *l-mers* is calculated. The *l-mer* with lowest distance is the consensus string. The algorithm is listed below:

Median_string_search()

```
{inputs: DNA,t,n,l
 output: bestword
 procedure:
 MedianStringSearch (DNA, t, n, l)
        bestWord ←AAA…A
        bestDistance ←∞
           for each l-mer s from AAA…A to TTT…T
                 if TotalDistance(s, DNA) < bestDistance
                    bestDistance←TotalDistance(s, DNA)
                    bestWord ← s
          return bestWord
}
```

In median string algorithm, all the 4^l combination of *l-mers* need to be examined. When the length of *l* increases, the number of candidate *l-mers* increase according to 4^l the execution time also increases. But all the 4^l *l-mers* are not significant. A significant amount of the generated 4^l *l-mers,* will not be the consensus. Even some candidate motifs can be found in no sequence. Those motifs got generated as a result of permutation and combination. For those motifs too, median string still perform calculation and contribute to the time complexity. In this paper, we introduced chapman-kolmogorov relation to exclude insignificant candidate motifs and reduce the time complexity of median string algorithm.

III. Chapman-Kolmogorov Relation

A time series of random variables is said to be a markov chain if it has the following property:

$$P(X_t = x_t | X_0 = x_0, X_1 = x_1, \ldots\ldots\ldots, X_{t-1} = x_{t-1}) \qquad (1)$$

It is said to be homogeneous or stationary if it also satisfies

$$P(X_t = z | X_{t-1} = y) = P(X_{t+k} = z | X_{t+z-1} = y) \forall k \geq 0 \quad (2)$$

In other words, the time series of random variable is markovian if the future value of the random variable depend only on the value of random variable has at the present time. The value of the random variable in the future must be independent of the values that it had in the past. We showed that one time series of random variables have these property their behavior can be represented by a graph like the one shown in Fig. 2.:

The possible values that the random variables can have are shown in circles, so in this example the random variables can have the values A, B and C. Instead of talking about the values that the random variables can have though, we state the circles represents the states the system can be in.

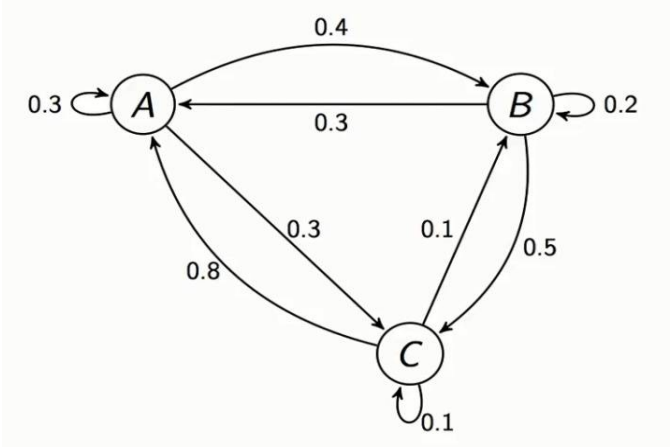

Fig. 2. Markov chain

The circles that represent states are connected by arrows and above these arrows numbers between 0 and 1 appear that represents the probability of moving between the states in a single time step. In this example the probability of moving from state c to state a is 0.8 in a single time step. Alternatively we might state the probability that a random variable has value c followed by a is equal to 0.8. From this markov chain, we can construct a transition probability matrix. We said that the rows in the matrix represent the initial states the system is in and the columns represents the states where the system moves at the next step. The position (a,b) in the matrix shows the probability of moving from state a to state b.

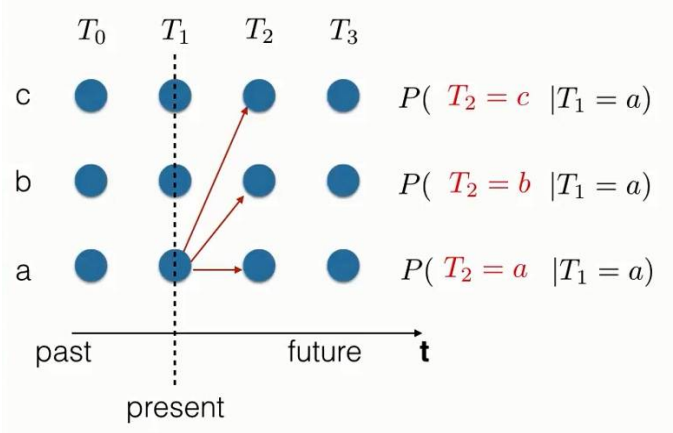

Fig. 3. One step transition

We now know how to obtain probability of T_2 being equal to a,b,c if T_1 =a. We can read this information either from the transition graph or from the one step transition probability matrix. What we would like now is to determine the conditional probability that T_3 takes on these three different values. In other words we would like to know, whether or not we can make prediction about the state of the system after two

or more time steps rather than after a single step. To see how, let us first consider the probability to the following happens in the future. We start with T_1=a, then move to T_2=c then again move to T_3=b. We know that we can get the probability of a single step from a to c from the transition matrix. Once we arrive in state c , i.e T_2=c, the system immediately forgets that T_1 was equal to a. The process is after all markovian and the state of the system was in the past do not affect the probability of the states in the future. Once we have moved from T_1=a to T_2=c, T_2=c becomes the present state of the system. As a consequence of this, we can also get the probability of the second transition T_2=c to T_3=b from the one step transition probability matrix. Because of the markov property, the event of moving from state T_1=a to T_2=c is independent of the event of moving from state T_2=c to T_3=b. The markov property ensures that the system immediately forget everything that happened in the past as soon as that is happened. Thus two adjacent transition are completely independent of each other. The only thing that affect the value of T_3 is the value of T_2. Now remember that we can obtain the probability that two independent event occur by multiplying the individual probability. Consequently for this particular problem, we can calculate the probability that the system will be moving from T_1=a to T_2=c then from T_2=c to T_3=b by multiplying the conditional probability T_3=b given T_2=c.

Let's now suppose, we want to calculate the probability for moving from state a to state b over two steps. There are three ways that can happen. We can go from state a to state c then on to state b. We can go from state a to state b then remain in state b. We can remain in state a for one step and then transfer to state b. The probability of each of the paths is shown in fig.

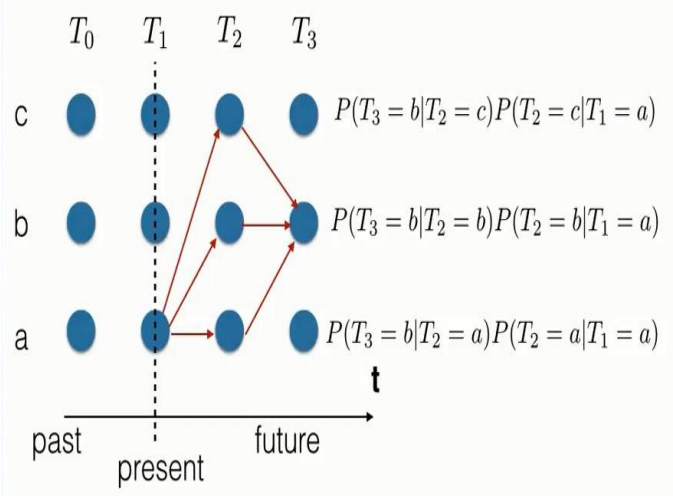

Fig. 4. Two step transition

Each of these pathway is mutually exclusive. As the random variable T_2 can only have one value, it cannot have two value simultaneously. We can thus obtain the total probability of moving from step a to step b over the course of two step by adding the probabilities of each of the paths. If we have only three steps, we can write everything explicitly. If we have more than three steps we can write that as a series notation like Eq

.

$$P(T_3 = b|T_1 = a) = \sum P(T_3 = b|T_2 = x_n)P(T_2 = x_n|T_1 = a) \quad (3)$$

This sum here rounds up all the possible value that the random variable T_2 can take. This sum of product is nothing more than the definition of the matrix product. Because the markov chain is homogeneous, the first matrix and second matrix are identical. We can thus obtain the two step probability matrix by multiplying the one step probability matrix by itself. In general,

$$A^{m+n} = A^m A^n \quad (4)$$

This is what is called chapman kolmogorov relation.

IV. Chapman Kolmogorov Relation Based Median String Algorithm

Consider a database with 10 data samples ($S1,S_2,\ldots\ldots,S_{10}$). Each sample is compose of 80 nucleotides.

TABLE I. DNA Data Set

SID	Sample
S_1	tagtggtcttttgagtgtagatctggagggaaagtatttccaccagttcggggtcacccagcaggcagggtgacttaat
S_2	cgcgactcggcgctcacagttatcgcacgtttagaccaaaacggagttggatccgaaactggagtttaatcggagtcctt
S_3	gttacttgtgagcctggttagacccgaaatataattgttggctgcatagcggagctgacatacgagtaggggaaatgcgt
S_4	aacatcaggctttgattaaacaatttaagcacgtaaatccgaattgacctggtgacaatacggaacatgccggctccggg
S_5	accaccggataggctggttattaggtccaaaaggtagtatcgtaataatggctcagccatgtcaatgtgcggcattccac
S_6	tagattcgaatcgatcgtgtttctccctctggtggttaacgaggggtccgaccttgctcgcatgtgccgaacttgtaccc
S_7	gaaatggttcggtgcgatatcaggccgttctcttaacttggcggtgcagatccgaacgtctctggaggggtcgtgcgcta
S_8	atgtatactagacattctaacgctcgcttattggcggagaccatttgctccactacaagaggctactggtgtgatccgta
S_9	ttcttacacccttctttagatccaaacctgttggcgccatcttcttttcgagtccttgtacctccatttgctctggtgac
S_{10}	ctacctatgtaaaacaacatctactaacgtagtccggtctttcctggtctgccctaacctacaggtcgatccgaaattcg

From this database we can construct a transition matrix like the following:

$$Transition = \begin{array}{c|cccc} & a & c & g & t \\ a & 49 & 39 & 46 & 50 \\ c & 48 & 46 & 39 & 56 \\ g & 40 & 48 & 58 & 53 \\ t & 48 & 55 & 56 & 59 \end{array}$$

Fig. 5. Transition matrix

The transition matrix contains the number of occurance of each of the sixteen possible digrams in the database. For

example Transistion(1,1)=49, that means there are 49 aa in the database. Similarly, Transition(4,4)=59, which means there are 59 tt in the data base. There are total 790 digrams in the database. Dividing all the elements of the matrix by sum of element, produces transition probability matrix like Fig. 6. A[i,j] represents the probability of transition from position *i* to position *j*.

```
A=      a           c           g           t
   a[0.06202532 0.06075949 0.05063291 0.06075949]
   c[0.04936709 0.05822785 0.06075949 0.06962025]
   g[0.05822785 0.04936709 0.07341772 0.07088608]
   t[0.06329114 0.07088608 0.06708861 0.07468354]
```

Fig. 6. Transition probability matrix

In this matrix A(1,1) indicates the probability of finding an a after a, A(2,3) indicates the probability of finding a g after c etc. If we multiply A by itself, we will have a matrix that indicates the probability of occurance of a particular nucleotide at the second step after a particular nucleotide. let us suppose, we are searching for 5 nucleotide sequence i.e our l-mer size is 5. In this case, if we multiply A by itself by 4 times, we will have a matrix like Fig. 7. This matrix contains the probability at fifth position.

```
A⁵=     a           c           g           t
   a[9.18004782e-07 8.28077016e-07 3.32785581e-07 8.28077016e-07]
   c[2.93215942e-07 6.69350669e-07 8.28077016e-07 1.63560335e-06]
   g[6.69350669e-07 2.93215942e-07 2.13306707e-06 1.78980072e-06]
   t[1.01558100e-06 1.78980072e-06 1.35907646e-06 2.32340330e-06]
```

Fig. 7. Transition probability matrix after five step.

This matrix indicates the probability of occurance of a particular nucleotide after 5 step of a particular nucleotide. For example, A(1,1) indicates the probability of occurance of an a after five step of an a. There are 16 elements in the matrix. Summing all the elements and then dividing that value by 16 we will get a threshold value. For the values that are greater than the threshold value, placing a 1 at that location and placing a 0 at other location we will have another matrix like Fig. 8. We can call this a significant matrix S. This matrix tells which transitions are significant and which transitions are not. The significant matrix contains 0's at some positions and 1's at some positions. 0's means insignificant, where as 1's mean significant.

```
S=     a  c  g  t
   a[0. 0. 0. 0.]
   c[0. 0. 0. 1.]
   g[0. 0. 1. 1.]
   t[0. 1. 1. 1.]
```

Fig. 8. Significant matrix

Having the significant matrix, we generate some rules acc ording to the significant matrix. For example, S(2,4)=1 means that the rule c→t is significant, S(3,4)=1 means rule g→t is sig nificant, S(2,3)=0 means rule c→c is insignificant etc. As per t he significant matrix, generated rules are listed in table II.

TABLE II. GENERATED RULES

First element (l_1)	Second element(l_2)	Count	Rule
c	t	56	c→t
g	g	58	g→g
g	t	53	g→t
t	c	55	t→c
t	g	56	t→g
t	t	59	t→t

From these rules, we can generate some motif of length 5. Any motif must start either from a or c or g or t. We will have four trees for motifs starting with four nucleotide. For exampl e one tree is depicted below in Fig. 9.

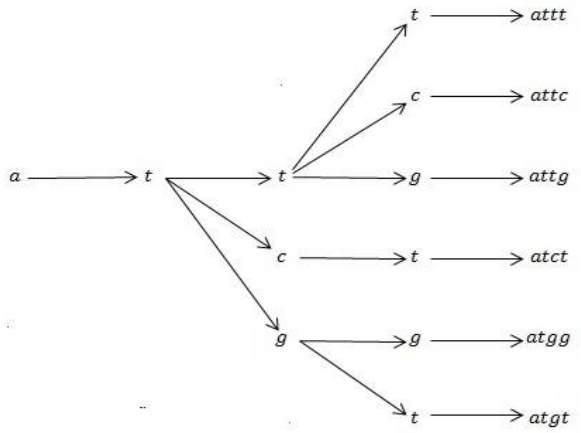

Fig. 9. Motif generation tree

From all the four trees, we will have a short list of motifs which is much shorter than 4^l. We will now use this short set of motifs as the search space of median string algorithm.

V. RESULT AND IMPLEMENTATION

To check the idea behind this concept, we have used Jupyter Notebook as Python programming language. The system configuration is:

- Processor: Intel core i7 CPU
- Clock rate:3.6 GHz
- HardDisk:1000GB
- RAM:8GB

Chapman Kolmogorov relation based median string algorithm accelerates the whole process easy and faster. We have compared the performance of Chapman Kolmogorov relation based median string algorithm with the performance of median string algorithm. In this demonstration we have compared the number of motifs generated by Chapman Kolmogorov relation based median string algorithm and median string algorithms for different *l-mer* size. Since our proposed system produces a only the significant motifs, for all l-mer size the number of motifs generated by the proposed system are found to be shorter than those produced by the

median string algorithms. The scenario is depicted in the table III.

TABLE III. NUMBER OF GENERATED MOTIFS

l-mer Size	Number of Motifs (Chapman Kolmogorov Relation Based)	Number of Motifs (Median String)
2	6	16
3	14	64
4	31	256
5	70	1024
6	157	4096
7	353	16784

The table narrates the results of both the methods used here. The results are the number of motifs generated by both the methods for different *l-mer* size. For *l-mer* size 2, Chapman Kolmogorov relation based system generates 6 motifs whereas Median string algorithm generates 16 motifs. So for *l-mer* size 2, Markov chain based system is (16-6)/6=1.66=166% better than Median string based system.

For *l-mer* size 3, Chapman Kolmogorov relation based system produces 14 motifs and Median string produces 64 motifs. Hence Chapman Kolmogorov relation based systems performance is (64-14)/14=3.57=357% better than median string performance. For *l-mer* size 5 , Chapman Kolmogorov relation produces 70 motifs and Median string produces 1024 motifs, as result Chapman Kolmogorov relation based system is (1024-70)/70=13.62=1362% better than Median string. For last data set the difference between two system reaches 4654%.So it is clear that, as the length of *l-mer* increases, Chapman Kolmogorov relation based system will produce comparatively more less number of motifs than the median string algorithm. The graphical analysis in Fig. 10. also reflects the same impact. The gray line is gradually in upward trend for Median String outcome and the blue line is the Chapman Kolmogorov relation based system outcome.

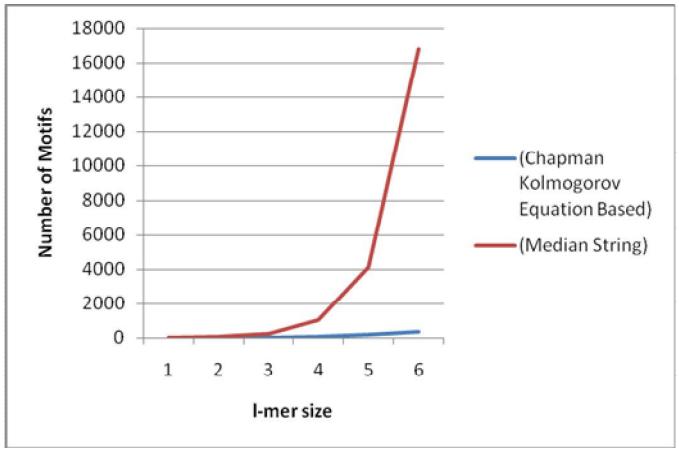

Fig. 10. Number of motifs generated

Same datasets have been imposed on core i7 computer with higher storage and RAM. In that case the processing time of Chapman Kolmogorov relation based system are also less than Median string based processing time.

We have checked every level time consumed by both methods. Median string method takes huge time for DNA consensus. Time consumed by these two methods convey their strengths

TABLE IV. TIME COMPARISION

l-mer Size	Chapman Kolmogorov relation Based Time(ms)	Median String Time(ms)
2	2.57	6.6
3	6.86	30.4
4	16.8	135
5	41.4	590
6	97.4	2590
7	236	11100

Every rows of the table IV demonstrates that Chapman Kolmogorov relation based Median string algorithm consumed less time than Median string orientation. For *l-mer* size 2, the time difference is (6.6-2.57)ms=4.03 ms. It is about 4.03/6.6=61% of the estimated time. That indicates that, Chapman Kolmogorov relation centric Median string finds DNA consensus faster than ordinary Median string algorithm. The time differences have been increased as the length of *l-mer* size increased. From the last row, Chapman Kolmogorov relation based analysis consumed 236 ms whereas ordinary median string algorithm consumed 11100 ms. There are 10864 ms difference for *l-mer* size 7. However it can be applicable for small *l-mer* size too. When *l-mer* size increases, we should consider Chapman Kolmogorov relation based system. In Fig. 11.we can see that, as the l-mer size increases, the time taken by the proposed system is much less than the time taken by the ordinary median string algorithm . Because the proposed system only produces significant motifs.

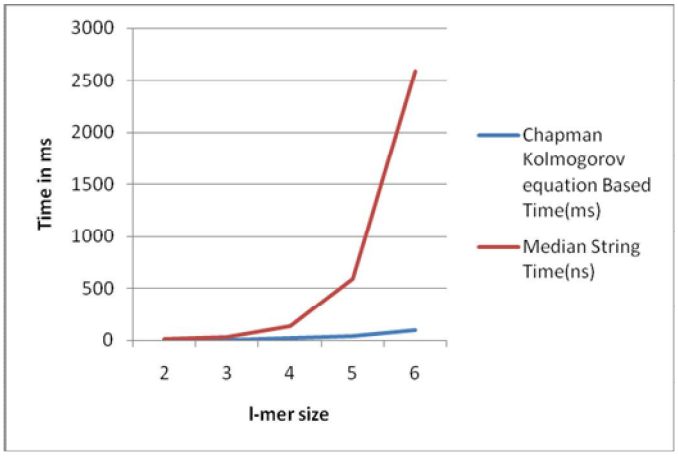

Fig. 11. Time comparision

VI. Conclusion

Median string algorithm generates all the 4^l number of l-mers, then fits each l-mer with the database and calculate the distance. In course of generating the l-mers, median string algorithm, generates significant and insignificant l-mers. Though this process takes much time but this is a certain process. Since we can generate a markov chain from the DNA data set, then also can find the chapman-kolmogorov relation. From this relation we generated a set of rules. Using that rules we produced a reduced set of significant motifs. Since the search space has been reduced, the proposed system takes much less time to find the consensus string. The way we have applied chapman-kolmogorov relation, there might be some other way of application. We believe that, further research may find some other way of application of chapman kolmogorov relation that may reduce the set of rules as well as number of generated motifs. We are working on that ground.

Reference

[1] M.Kellis, N.Patterson, B. Birren, B.Berger, and E. S. Lander, "Methods in comparative genomics: Genome correspondence, gene identification and regulatory motif discovery". *Journal of Computational Biology*,Vol.11,pp.319–355,2004.

[2] J. D.Thompson,D. G. Higgins, and T. J.Gibson,"CLUSTAL W:Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice". *Nucleic Acids Research,* Vol.22, pp. 4673–4680,1994.

[3] T. D. Schneider, "Consensus sequence Zen". *Applied Bioinformatics,* Vol. 1. pp. 111–119, 2002.

[4] C.Lawrence, S.Altschul, M.Boguski, J.Liu, A.Neuwald, and J.Wootton , "Detecting subtle sequence signals: A gibb"s sampling strategy for multiple alignment". *Science*, Vol. 262, pp. 208–214, 1993.

[5] T.Bailey, and C.Elkan, "Fitting a mixture model by expectation maximization to discover motifs in biopolymers". *2nd International Conference on Intelligent Systems for Molecular Biology,* 14-17 August,Stanford,CA, pp. 28–36, 1994.

[6] Y.Zhang, H.Huo, and Q.Yu, "A heuristic cluster-based em algorithm for the planted (l, d) problem". *Journal of Bioinformatics and Computational Biology*, Vol. 11, no. 4, pp. 1350009, 2013.

[7] P.Kuksa, and V.Pavlovic, "Efficient motif finding algorithms for large-alphabet inputs". *BMC Bioinformatics,* Vol. 11, no. Suppl 8, p. S1, 2010.

[8] S.Altschul, and D.Lipman, "Trees, stars, and multiple sequence alignment". *SIAM Journal on Applied Mathematics*, Vol.49, pp.197–209, 1989.

[9] J.Gramm, F.Hüffner, and R.Niedermeier, "Closest strings, primer design, and motif search". *Proceedings of the Sixth Annual International Conference on Computational Biology, RECOMB 2002,*18-21 April, Washington, DC, USA,pp. 74–75, 2002.

[10] J.Gramm, R.Niedermeier, and P.Rossmanith, "Exact solutions for closest string and related problems". *Proceedings of the 12th International Symposium on Algorithms and Computation,* 19-21 December,Christchurch,Newzealand, pp. 441–453, 2001.

[11] R.M.Karp, "Mapping the genome: some combinatorial problems arising in molecular biology".*Proceedings of the 25th Annual ACM Symposium on Theory of Computing,* 16-18 May, San Diego,CA,USA,pp. 278–285, 2001.

[12] M.Li, B.Ma, and L.Wang, "On the closest string and substring problems". *Journal of the ACM,* Vol.49, no.2, pp.157–171, 2002.

[13] H.Mauch, M.J.Melzer, and J.S.Hu, "Genetic algorithm approach for the closest string problem". *Proceedings of the 2nd IEEE Computer Society Bioinformatics Conference,*pp. 560–561, 2003.

[14] C.N.Meneses, Z.Lu, C.A.S.Oliveira, and P.M. Pardalos, "Optimal solutions for the closest-string problem via integer programming". *INFORMS Journal on Computing,* Vol.16, no.4, pp.419–429, 2004.

[15] F.Nicolas, and E.rivals, "Complexities of the centre and median string problems". *Proceedings of the 14th Symposium on Combinatorial Pattern Matching,* 25-27 June, Michoacan , Mexico, pp. 315–327, 2003.

[16] J.Gramm, R.Niedermeier, and P.Rossmanith, "Fixed-parameter algorithms for closest string and related problems". *Algorithmica*, Vol.37, no.1, pp.25–42, 2003.

[17] B.Ma, and X.Sun, "More efficient algorithms for closest string and substring problems". *Proceedings of the 12th Annual International Conference on Research in Computational Molecular Biology,*30 March-2 April, Singapore, pp. 396–409, 2008.

[18] N.Stojanovic, P.Berman, D.Gumucio, R.Hardison, and W.Miller, "A linear-time algorithm for the 1-mismatch problem". *Proceedings of the 5th International Workshop on Algorithms and Data Structures,* 6-8 August, Nova Scotia, Canada,pp. 126–135, 1997.

[19] A.Ben-Dor, G.Lancia, J.Perone, and R.Ravi, "Banishing bias from consensus sequences".*Proceedings of the 8th Symposium on Combinatorial Pattern Matching,* Aarhus,Denmark, pp. 247–261, 1997.

[20] L.Gasieniec, J.Jansson, and A.Lingas, "Efficient approximation algorithms for the Hamming center problem". *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms,* 17-19 January, Baltimore, MD, USA, pp. 905–906,1999.

[21] L.Gasieniec, J.Jansson, and A.Lingas, "Approximation algorithms for Hamming clustering problems". *Journal of Discrete Algorithms*, Vol.2, no.2, pp.289–301, (2004).

[22] K.Lanctot, M.Li, B.Ma, S.Wang, and L.Zhang, "Distinguishing string selection problems".*Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms,* 17-19 January, Baltimore, MD, USA, pp. 633–642, 1999.

[23] M.Li, B.Ma, and L. Wang, "Finding similar regions in many strings". *Proceedings of the 31st Annual ACM Symposium on Theory of Computing,* pp. 473–482, 1999.

[24] C.Boucher, D.Brown, and S.Durocher, "On the structure of small motif recognition instances".*Proceedings of the 15th Symposium on String Processing and Information Retrieval,* pp. 269–281, 2008.

[25] S.Sze, S.Lu, and J.Chen, "Integrating sample-driven and pattern-driven approaches in motif finding". *Proceedings of the 4th Workshop on Algorithms in Bioinformatics,* pp. 438–449, 2004.