Current Research Journal of Biological Sciences 4(4): 394-400, 2012
ISSN: 2041-0778

Submitted: January 11, 2012 Accepted: February 17, 2012 Published: July 10, 2012

Discovered DNAs of Protein with Using Parallel Prefixspan Method

[1]Mehdi Sobhkhiz Talouki and [2]Reza Noorian Talouki
[1]Sama Technical and Vocational Training College, Islamic Azad University, Sari Branch, Sari, Iran
[2]Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

Abstract: Discovery of Sequential pattern mining is an important data mining mission with wide applications. There is no model that used multi coring techniques for parallel mining of closed sequential patterns. The parallelization of a prefixspan method to discover DNAs is proposed in this study. The prefixspan method is used to extract the frequent pattern from a sequence database. This system requires the use of multiple computers connected in local area network. This algorithm includes multi-coring to achieve communication between a master process and multiple slave processes. This algorithm applies dynamic scheduling to avoid tasks idling. Moreover we employ a technique, called selective sampling. We implement this algorithm with using a 4G memory and AMD phenomX4. Our experimental results show that this algorithm attains good efficiencies on motifs extraction.

Keywords: Cores scheduling, DNAs discovery, DNA sequences, sequence mining, task scheduling

INTRODUCTION

The objective of sequential pattern mining is to discover frequent subsequences in a dataset. Sequential pattern mining has multiple applications, comprising the finding frequent units in DNA sequences, the inspection of scientific or curative processes and analysis of web log registered acts. a lot of sequential pattern mining algorithms have been proposed bygone. (Jian *et al*., 2007; De Amo *et al*., 2008). Since a long sequence includes a combinatorial number of subsequences, sequential pattern mining creates an explosive number of frequent subsequences for long patterns, which is preventively costly in both time and space (Yan *et al*., 2003a; Han and Kamber, 2006). Hence, instead of mining the whole set of sequential patterns, an alternative but equally powerful explanation is to mine closed sequential patterns only (Yan *et al*., 2003b; Han and Kamber, 2006).

A motif is a featured pattern in amino acid sequences. It is related to a function of the protein that has been preserved in the evolutionary process of an organism. The amino acid sequence is composed of 20 kinds of alphabets. The featured pattern, which includes wild cards, is discovered from frequent patterns extracted in the sequences. A Protein sequence motif, signature or consensus pattern is a short sequence that is embedded within the sequences of a same protein family (Bork and Koonin, 1996). By identifying protein sequence motifs, an unknown sequence can be quickly classified into its computationally predicted protein family/families for further biological analysis.

In past years, many algorithms for finding protein sequence motifs have been proposed. Sequence motif discovery algorithms can be generally categorized into 3 types:

- String alignment algorithms
- Exhaustive enumeration algorithms
- Heuristic methods

String alignment algorithms (Waterman *et al*., 1984; Delcoigne and Hansen, 1975; Needleman and Wunsch, 1970) find sequence motifs by minimizing a cost function which is related to the edit distances between sequences. Multiple alignments of sequences are a NP-hard problem and its computational time increases exponentially with the sequence size. Local search algorithms such as Gibbs sampling (Lawrence *et al*., 1993) expectation maximization (Bailey and Elkan, 1995a, b) may end up in a local optimum instead of finding the best motif (Buhler and Tompa, 2001). Exhaustive enumeration algorithms (Blanchette *et al*., 2000; Brazma *et al*., 1998) are guaranteed to find the optimal motif, but run in exponential time with respect to the length of motif. Heuristic methods (Inge *et al*., 1995; Sagot and Viari, 1996) can have a better performance but are usually less flexible.

The existing other pattern extraction algorithms, which include multiple alignment, statistical method,

Corresponding Author: Reza Noorian Talouki, Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

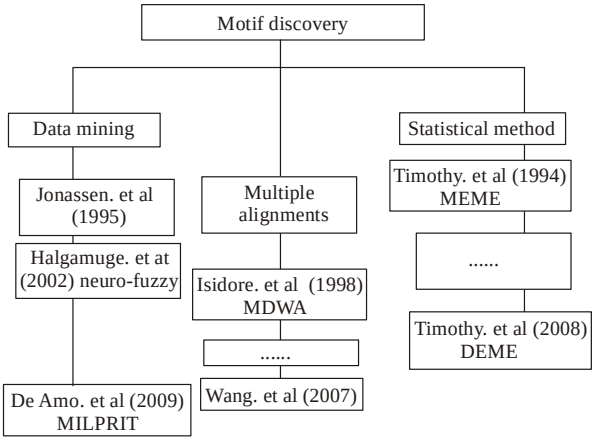


Fig. 1: Previous studies about motif discovery

present some problems (Hajime *et al*., 2002; Chao and Srinivasan, 2004; Zeeshan *et al*., 2010; Zhiping *et al*., 2004; Chang *et al*., 2006). These algorithms are neither functional nor fast for the discovery of motifs from large-scale amino-acid sequences.

To solve the functional problem, we apply some changes on prefixspan method, (Chang *et al*., 2006) which introduce a mechanism to limit the number of wildcards to the prefixspan method. Using this modifications was successful in the deletion of extra patterns and in the reduction of computational time for about 1200 amino acid sequences, each of which ranges in length from 20 to 3800 characters. To achieve faster pattern extraction from large-scale sequence databases, the parallelization of the Prefixspan method by the use of multiple computers connected in local area network was used and is explained in this paper. To achieve better parallelization we used a dynamic load balancing method. (Shengnan *et al*., 2004; Giuseppe and Michael, 2006). One computer, which manages parallel processing, divides the entire extraction into multiple k-length frequent-pattern extractions. The k-length frequent pattern extraction is distributed to the computers, which perform parallel processing. As a feature of the Prefixspan method, the workload of each k-length frequent pattern extraction processing is different. After a k-length frequent pattern is extracted by a computer, the step is repeated. Other computers continue processing, one after another, while a designated computer extracts a k-length frequent pattern.

Previous studies about motif discovery: In past years, many algorithms for finding protein sequence motifs have been proposed. Sequence motif discovery algorithms can be generally categorized into 3 types:

- String Alignment algorithms
- Exhaustive enumeration algorithms
- Heuristic methods

String alignment algorithms (Waterman *et al*., 1984; Delcoigne and Hansen, 1975; Needleman and Wunsch, 1970) find sequence motifs by minimizing a cost function which is related to the edit distances between sequences. Multiple alignments of sequences are a NP-hard problem and its computational time increases exponentially with the sequence size. Heuristic methods (Inge *et al*., 1995; Sagot and Viari, 1996) can have a better performance but are usually less flexible.

The existing pattern extraction algorithms, which include multiple alignment, statistical method, present some problems (Hajime *et al*., 2002; Isidore and Aris, 1998; Chao and Srinivasan, 2004; Erik *et al*., 1997; Zeeshan *et al*., 2010). These algorithms are neither functional nor fast for the discovery of motifs from large-scale amino-acid sequences.

There are a lot of algorithms for solving this problem but none of them are as useful as our algorithm (Chao and Srinivasan, 2004; De Amo *et al*., 2008). In Fig. 1 shown previous studies about motif discovery. Our represented method for motifs extraction is suitable than explained methods. Our algorithm also can be suitable to find all of the frequent pattern. In this method it is not necessary to specify the average length of sequences.

METHODOLOGY

Prefixspan method: In here, we present a brief description of this algorithm. Let DB be a sequence dataset (Table 1). The algorithm starts with a scan of DB to identify the frequent 1-sequences. Then, a second scan of DB constructs the projected datasets for the frequent

Table 1: An example of sequence dataset

Sequence id	Sequence
10	CAABC
20	ABCB
30	CABC
40	ABBCA

Table. 2: Dataset for example

Seq_id	Sequence
1	MFKALRTIPVILNMNKDSKLCPN
2	MSPNPTNHTGKTLR

1-sequences. Let i be a sequence, a projection i of DB, denoted as P(i, DB), is a set of subsequences, which are made up of the sequences in DB containing i after deleting the events appearing before the first occurrences of i within each sequence for instance, Table 1 shows a sequence dataset, With the support threshold as 2 the projected dataset for sequence AB is P(AB, DB) = {C, CB, C, BCA} as you see sequences of AB routes be deleted and remain subsequence constitute this set (Han and Kamber, 2006; Jian *et al*., 2001). Performance studies (Jian *et al*., 2007; Zhiping *et al*., 2004) have shown that the prefixspan algorithms is more efficient than the other algorithm (Zaki, 2001a, b, c; Yan *et al*., 2003c; Agrawal and Srikant, 1995a, b).

After the projected datasets are built, algorithm searches each projected dataset and selects the sequential patterns. The existing prefixspan method cannot generate frequent patterns including some wildcards. For instance, P = ATT***TTC*GGATT**T*T*CCC*GG" are considered to be a protein sequence. Where * indicates one wild card symbol. It can be every symbol. The common prefixspan method considered TT***T and TT**T as a TTT pattern, but in our suggested algorithm they known as TT2T and TT3T. The prefixspan method generates (k+1)-length frequent patterns from each k-length frequent pattern in a set of sequences. The last character of each (k+1)-length frequent pattern is found from one of the characters that exist among the next position of a k-length frequent pattern and the last position in the sequences.

First of all, the Prefixspan method extracts the 1-length frequent patterns with considering their minsup. This method extracts the 2-length frequent patterns from one 1-length frequent pattern. In fact, this method extracts the (k+1)-length frequent pattern from k-length frequent pattern. For instance, Fig. 1 shows frequent patterns that are extracted in the two sequences, the number of wildcards is 3 (Table 2). Figure 1 show how this represented algorithm work. This method extracts 1-length frequent patterns, "K, L, M, N, P, R, S, T" that support minsup. Next, this method extracts 2-length frequent patterns from 1-length frequent patterns, When the 1-length frequent pattern is "K" the 2-length frequent patterns are "K*L".

Because of the different number of wildcards in the two sequences, the 2-length frequent pattern,"K1L" is extracted. Next, this method extracts 3-length frequent patterns from the 2-length frequent pattern, "K*L". The extracted 3-length frequent pattern is "K*LR" when the 1-length frequent pattern is "M" the 2-length frequent patterns are "MN". Because of the different number of wildcards in the two sequences, the 2-length frequent pattern, "MN" is not extracted. The 2-length frequent pattern "MS" is also not extracted because of the different number of wildcards. Thus this algorithm use Prefixspan to extract wildcards and number of them.

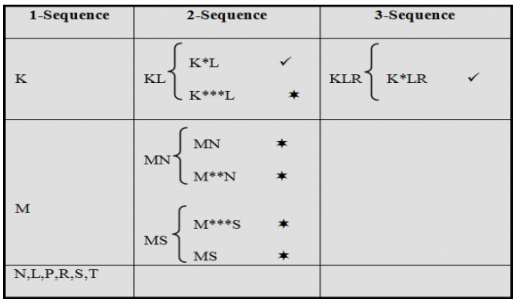


Fig. 2: Prefixspan method

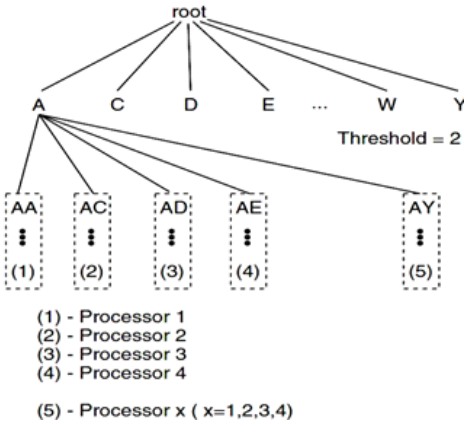


Fig. 3: Parallel tree search for min sup = 2 [36]

Parallel frequent pattern extraction in prefixspan method:

Our algorithm follows three steps:

- **Step 1:** Identify the frequent 1-sequences.
- **Step 2:** Project the dataset along each frequent 1-sequence.
- **Step 3:** Mine each resulting projected dataset.

The projected datasets of the frequent 1-sequences are independent. Given a 1-sequence, say i, only the suffixes that follow the first occurrences of i in each sequence are the projection of the dataset along i. Therefore, the closed sequential-patterns mined from the dataset projection along i_1 all start with i_1 as the prefix while the patterns discovered from i_2's projections all start with i_2. A partition strategy like the one just described is

convenient for task decomposition. Since the projected datasets are independent, they can be assigned to different thread. Then, each thread can mine the assigned projected data sets independently. No inter-processor communication is needed during the local mining. Work procedure of algorithm when minsup = 2 and thread = 4 shown in Fig. 2. (Nodes of each sub tree are called jobs). Amino acids correspond to the letters "A, C, D, E, F, G, H, I, K, L, M, N, P,Q, R, S, T, V, W, Y" of the alphabet.

In Fig. 3, because the minsup = 2, the combination of alphabets of the amino acid shows 400 jobs, which are 20 letters times 20 letters. When the jobs are extracted, for instance, with four computers, each computer will do 100 jobs on the average. Our strategy for the parallel mining of patterns is as follows:

Each thread counts the occurrence of 1-sequences in a different part of the dataset.

For each frequent 1-sequence a very compact representation of the dataset projections, called pseudo-projections, is built. This is done in parallel by assigning a different part of the dataset to each thread. In fact preparation of pseudo-projections from purposed data set and abandon of data set to each thread both done in the same time.

Dynamic scheduler distributes the projection across the thread for processing.

Algorithm GapPPS(I, DB,min_sup,M,N)
Input:(1) I is the Processor ID, (2) DB is a portion of the dataset
assigned to processor I, (3) min_sup is the minimum support
threshold, (4) M and N: the parameters of a gap constraint.
Output: the set sequence Proteins with Wild Card.

I_1: Find the set of length-1frequent sequential patterns, L_1
I_2: PSP = projection(L_1,DB)
I_3: GPSP = broadcast (PSP)
I_4: S = selective sampling(L_1,I,DB,min_sup,M,N)
I_5: L_2 = partition(L_1,S);//the new set of projections to L_2
I_6: if (I == 0) then
I_7: Accept requests from slave nodes and reply to each; Request with a new identifier from set L_2
I_8: Else
I_9: Send request for a projection identifier to the master node
I_{10}: Stop if all projections have been assigned
I_{11}: Apply algorithm to element of GPSP assigned by the master processor
I_{12}: Enter patterns into output set send request operation
I_{13}: End if

Our Algorithm

Threads scheduling: Next, we discuss the mechanism that we use to assign projections to thread. To reduce load imbalance thread Prefixspan uses dynamic scheduling. In our implementation, there is a master thread which maintains a queue of pseudo-projection identifiers. Each of the other thread is initially assigned a projection. After a thread completes the mining of a projection, it sends a request to the master thread for another projection. The master thread replies with the index of the next projection in the queue and removes it from the queue. This process continues until the queue of projections is empty. The requests and replies are short messages and, therefore, the request and reply time is usually negligible relative to the mining time. Dynamic scheduling is quite effective when the sub thread is of similar size and the number of threads is equal with the number of processors.

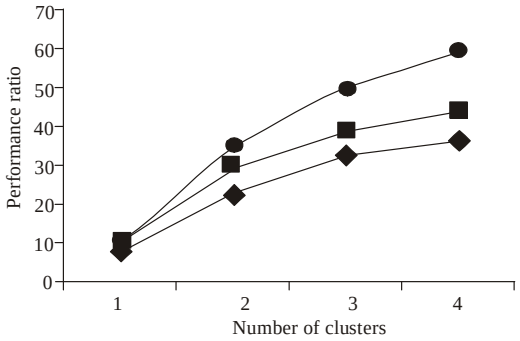
For the datasets we used in our experiments, the cost of mining the projections may very greatly. The relatively large mining time of some projected datasets may result in extremely imbalanced workload.

Implementation: We surmise that the database is divided into N subsets and that the subset allotted to core I is denoted DB_I. This entire act is performed to compute the global counts of the frequent1-sequences are identified and stored into variable L_1. Next, each core builds pseudo-projections for the frequent 1-sequences within the allotted portion of the dataset. The pseudo-projections are broadcast to the entire core. Before scheduling the projections, MCSP applies selective sampling to appraise the relative mining time of these projections.

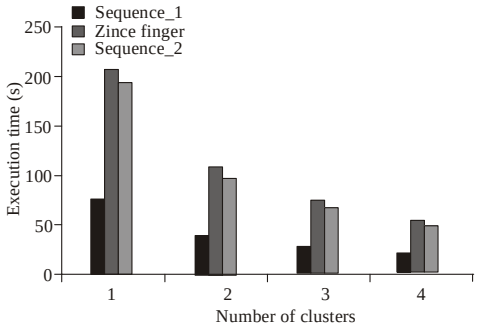
The function selective sampling () implements the process of mining the a selective sample. But in place of producing the closed sequential patterns, it records the mining time for the projections of all frequent 1-sequences. Variable S is allotted these relative mining times. After the sampling, the top time-consuming projections are divided into smaller ones. For example, if the projection along A is one of the top time consuming projections, it will be divided into the projections along zy, zx and so on. Then the master core schedules these projections as subtasks by preserving a task queue. The projections appraised to take longer time in sampling are to be scheduled earlier. Those very small projections can be scheduled in pieces to prevent communication contention. Core0 is treated as the master core and taking charge of the task scheduling while the other entire core (slave core) mine the allotted projections independently without communication to each other. Whenever a slave core finishes the allotted subtask, it sends a request to the master core for another one, until the task queue is vacant. Each slave core outputs the closed sequential patterns in a file. The total closed sequential patterns are the concatenation of these files.

Table 3: Detail of used datasets

	#Seq	Total.len	Ave. seq. len	Max. seq. len
Sequence_1	110	33385	435	3872
Zince finger	467	245595	525	4036
Sequence_2	320	205220	495	4025



(a)



(b)

Fig. 4: Execution time (A), Performance ratio (B)

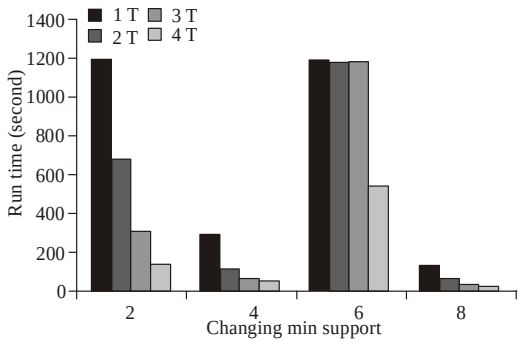


Fig. 5: Influence of changing minimum support

RESULTS AND ANALYSIS

- **Evolution:** Our performance study includes both artificial and actual datasets. We used three artificial datasets generated by the genbank dataset available in NCBI and an actual dataset zince finger, which comes from protein sequence set and provided by NCBI (www.ncbi.nlm.nih.gov/guide/dna-rna). In genbank, we consider distinctive products as distinctive items and the sequence views as events. Also we assume all of the consecutive sequences of DNAs as a protein sequences. The characteristics of these datasets are shown in Table 3.

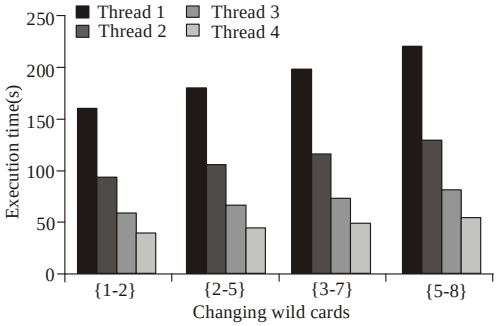


Fig. 6: Influence of changing wild cards

Figure 4 shows the execution time and the performance ratio in when the number of processor is increased from two to eight. The performance ratio of each parameter is as follows:

- Zince finger data set, minsup = 2, minimum support ratio 40 and wild cards number 2-7.
- Sequence_1 data set, minsup = 2, minimum support ratio 40 and wild cards number 2-5.
- Sequence_2 data set, minsup = 2, minimum support ratio 40 and wild cards number 2-7.

Figure 4 shows the total execution time and speed increasing for each data set. In this paper, execution time measured by second. From the result in Fig. 4B, the performance ratio is about 2 times when 2 computers are used, 2.6 times when 3 computers are used and 3.5 times when the maximum of 4 computers is used. When the parameters, which are the support ratio, wildcards and threshold, were changed, there was at greater difference at the execution time of each slave process can be attributed to the jobs.

Because of this the execution time of each job is different. If we can estimate the size of jobs, we expect that better the performance achieved. We tested the influence of changing minimum support threshold on the performance of our algorithm. The results are shown in Fig. 5. Our algorithm shows stable performance with different support threshold. Also we tested the influence of changing wildcards number on the performance of our algorithm. The results are shown in Fig. 6. As the result our algorithm shows stable performance with the changing wildcards number.

CONCLUSION

In this study, we propose a parallel mining algorithm. It can mine DNAs sequences with considering the wildcards. We apply multi coring for patterns mining. We exploit the divide-and-conquer feature to minimize the inter-processor communications. The PC cluster appeared to be effective, according to the results of the verification experiment. The amino acid sequence used by the verification experiment was a small-scale sequence.

REFERENCES

Agrawal, R. and R. Srikant, 1995a. Mining Sequential Patterns. In International Conference on Data Engineering (ICDE), IEEE Press, Taipei, Taiwan, pp: 3-14.

Agrawal, R. and R. Srikant, 1995b. Mining sequential patterns. In International Conference on Data Engineering (ICDE), IEEE Press, Taipei, Taiwan, pp: 3-14.

Bailey, T. and C. Elkan, 1995a. Unsupervised learning of multiple motifs in biopolymers using expectation maximisation. Mach. Learn., 21(1-2): 51-80.

Bailey, T.L. and C. Elkan, 1995b. The value of prior knowledge in discovering motifs with MEME. Processing International Conference Intelligence Systems Molecular Biology, 3: 21-29.

Blanchette, M., B. Schwikowski and M. Tompa, 2000. An exact algorithm to identify motifs in orthologous sequences from multiple species. Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, San Diego, pp: 37-45.

Bork, P. and E. Koonin, 1996. Protein sequence motifs. Curr. Opin. Struct. Biol., 6: 366-376.

Brazma, A., I. Jonassen, J. Vilo and E. Ukkonen, 1998. Predicting gene regulatory elements in silico on a genomic scale. Genomic Res., 15: 1202-1215.

Buhler, J. and M. Tompa, 2001. Finding Motifs using Random Projections. Proceedings of RECOMB, Montreal, Canada, pp: 69-76.

Chang, D.T., Y.Z. Weng, J.H. Lin, M.J. Hwang and Y.J. Oyang, 2006. Protemot: Prediction of protein binding sites with automatically extracted geometrical templates. Nucleic Acids Res., 34: 303-309.

Chao, W. and P. Srinivasan, 2004. Parallel Algorithms for Mining Frequent Structural Motifs in Scientific Data. Proceedings of the 18th annual international conference on Supercomputing, pp: 31-40.

Cong, S. and J. Han and D. Padua, 2004. Parallel mining of closed sequential Patterns. Parallel Comput., 30(4): 443-472.

De Amo, S. Giacometti, A. Pereira and W. Clemente, 2008. MILPRIT: A constraint based algorithm for mining temporal relational patterns. Int. J. Data Warehousing Mining, 4(4): 42-61.

Delcoigne, A. and P. Hansen, 1975. Sequence comparison by dynamic programming. Biometrika, 62: 661-664.

Erik, L.L.S., R.E. Sean and D. Richard, 1997. Pfam: A comprehensive database of proteins domain families based on seed alignments. Proteins, 28: 405-420.

Giuseppe, D.F. and R.B. Michael, 2006. Dynamic load balancing for the distributed mining of molecular structures. IEEE T. Parall. Distr., 17(8): 773-785.

Hajime, K., K. Tomoki and M. Yasuma, K. Susumu and Y. Yukiko, 2002. Modified Prefix Span method for Motif Discovery in Sequence Databases. PRICAI, Springer-Verlag, 2417: 482-491.

Han, J. and J. Pei, 2000. Mining frequent patterns by pattern-growth: Methodology and implications. ACM SIGKDD Explorations Newsexplainter, 2(2): 14-20.

Han, J. and M. Kamber, 2006. Data Mining Concepts and Techniques. 2nd Edn., Elsevierdirect, pp: 498-505.

Inge, J., F.C., John and G.H. Desmond, 1995. Finding flexible patterns in unaligned protein sequences. Protein Sci., 4(8): 1587-1595.

Isidore, R. and F. Aris, 1998. Motif discovery without alignment or enumeration. Proceedings of Second Annual ACM International Conference on Computational Molecular Biology (RECOMB), pp: 221-227.

Jian, P., H. Jiawei and W. Wei, 2007. Constraint-based sequential pattern mining: The pattern-growth methods. J. Intelligent Inf. Syst., 22(2): 133-160.

Jian, P., H. Jiawei, M.A. Behzad and P. Helen, 2001. prefix Span: mining sequential patterns efficiently by prefix-projected pattern growth. In International Conference on Data Engineering (ICDE), IEEE Computer Society Press, 16(11): 215-224.

Lawrence, C.E., S.F. Altschul, M.S. Boguski, J.S Liu, A.F. Neuwald and J.C. Wootton, 1993. Detecting subtle sequence signals: A Gibbs sampling strategy for muttiple alignment. Science, 262(5131): 208-214.

Needleman, S and C. Wunsch, 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Molecular Biol., 48: 443-453.

Sagot, M. and A. Viari, 1996. A double combinatorial approach to discovering patterns in biological sequences. Proceedings of the 7th Symposium on Combinatorial Pattern Matching, pp: 186-120.

Shengnan, C., H. Jiawei and P. David, 2004. Parallel mining of closed sequential patterns. Parallel Comput., 30(4): 443-472.

Waterman, M., D. Galas and R. Arratia, 1984. Pattern recognition in several sequences: Consensus and alignment. B. Math. Biol., 46: 512-527.

Yan, X., J. Han and R. Afshar, 2003a. Clospan: Mining closed sequential patterns in large datasets. In Intelligent Conferance Data Mining (SDM), pp: 166-177.

Yan, X. and J. Han and R. Afshar, 2003b. CloSpan: Mining closed sequential patterns in large datasets. Proceedings of the 3rd SIAM International Conference on Data Mining, pp: 166-177.

Yan, X. and J. Han and R. Afshar, 2003c. Clospan: Mining closed sequential patterns in large datasets. In Intelligent Conference Data Mining (SDM), pp: 166-177.

Zaki, M.J., 2001a. Spade: An efficient algorithm for mining frequent sequences. Mach. Learn., 42(1-2): 31-60.

Zaki, M.J., 2007b. Spade: An efficient algorithm for mining frequent sequences. Mach. Learn., 42(1-2): 31-60.

Zaki, M.J., 2001c. Parallel sequence mining on shared memory machines. J. Parallel Distr. Com., 61(3): 401-426.

Zeeshan, S., S. Collin, K. Manolis, I. Piotr and G. John, 2010. Motif discovery in physiological datasets: A methodology for inferring predictive elements. Trans. Knowledge Discovery Data (TKDD), 4(1): Article 2.

Zhiping, W., D. Mehmet and K. Sun, 2004. Guiding motif discovery by iterative pattern refinement. Proceedings of the ACM Symposium on Applied Computing, pp: 162-166.