A Grammatical Inference Sequential Mining Algorithm for Protein Fold Recognition

Taysir Hassan A. Soliman
Associate Professor
Information Systems Dept.
Faculty of Computers & Information
Assiut University, Egypt

Ahmed Sharaf Eldin
Professor
Information Systems Dept.
Faculty of Computers & Information
Helwan University, Egypt

Marwa M. Ghareeb
Lecturer
Information Systems Dept.
Faculty of Computers & Information
Modern Academy, Egypt

Mohammed E. Marie
Lecturer
Information Systems Dept.
Faculty of Computers & Information
Helwan University, Egypt

***Abstract*—Protein fold recognition plays an important role in computational protein analysis since it can determine protein function whose structure is unknown. In this paper, a Classified Sequential Pattern mining technique for Protein Fold Recognition (CSPF) is proposed. CSPF technique consists of two main phases: the sequential mining pattern phase and the fold recognition phase. In the sequential mining pattern phase, Mix & Test algorithm is developed based on Grammatical Inference, which is used as a training phase. Mix & Test algorithm minimizes I/O costs by one database scan, discovers subsequence combinations directly from sequences in memory without searching the whole sequence file, has no database projection, handles gaps, and works with variant length sequences without having to align them. In addition, a parallelized version of Mix & Test algorithm is applied to speed up Mix & Test algorithm performance. In the fold recognition phase, unknown protein folds are predicted via a proposed testing function. To test the performance, 36 SCOP protein folds are used, where the accuracy rate is 75.84% for training data and 59.7% for testing data.**

***Keywords*—Data mining; grammatical inference; sequential mining; protein fold recognition**

I. INTRODUCTION

Protein fold recognition is an important step towards understanding protein three-dimensional structures and their biological functions. Fold recognition techniques do not require similar sequences in the protein databank, just similar folds. Successful approaches have been applied to protein fold recognition [1]. For example, various researchers used Neural networks to predict protein folds, such as GeneThreader [2], TUNE (Threading Using Neural nEtwork) [3], neural networks with tailored early-stopping [4], Bayesian Networks [5], structural- pattern based methods [6], and Genetic Algorithms [7,8]. Examples of using Support Vector Machines (SVM) have been illustrated as follows: directly predict the alignment accuracy of a sequence template alignment [9] and a combined technique of Support Vector Machine (SVM) classifier with Regularized Discriminant Analysis (RDA) [10]. Other research has been performed using Monte Carlo methods [11]. In addition, many researchers used parallel evolutionary algorithms for protein fold recognition, such as parallel EST, probabilistic roadmap for motion planning, pRNAPredict for RNA secondary structure [12-16]. However, although significant improvement has been made, the accuracy of the existing methods remains low and there is a need for new methods contributing to the field of fold recognition.

Sequential mining algorithms have been proposed to predict protein folds. The objective of sequential pattern mining is to discover interesting sequential patterns in a sequence database. It is one of the essential data mining tasks widely used in many applications, including customer purchase pattern analysis and biological data sequences [17-22], etc. Many research have been performed to efficient sequential pattern mining, such as [23-25], closed and maximal sequential pattern mining [26-29], constraint-based sequential pattern mining [30-32] approximate sequential pattern mining [33], sequential pattern mining in multiple data sources [34], sequential pattern mining in noisy data [35], incremental mining of sequential patterns [36], and time-interval weighted sequential pattern mining [37]. Two of the general sequential mining algorithms are SPADE [24] and PrefixSpan [23], which are more efficient than others in terms of processing time. SPADE is one of the vertical-format based algorithms and uses equivalence classes in the mining process. PrefixSpan is one of the pattern-growth approaches. It recursively projects a sequence database into a set of smaller projected sequence databases and grows sequential patterns in each projected database by exploring only the locally frequent fragments. cSPADE [38] algorithm is a straightforward extension of SPADE algorithm. The only difference is the involvement of constraints in the cSPADE. These constraints include length, width, and duration limitations on the sequences, item constraints, event constraints, and incorporating class information. In addition, one of the SPADE based algorithm called SPAM (Sequential PAttern Mining) [39] has been proposed. It integrates the ideas of GSP, SPADE, and

FreeSpan and combines a vertical bitmap representation of the database with efficient support counting.

One of the promising areas is Formal Language Theory and Grammatical Inference (GI), which is playing important role in the development of new methods to process biological data [40]. Many works propose GI techniques to tackle bioinformatics tasks, such as secondary structure identification [41], protein motifs detection [42], and optimal consensus sequence discovery [43]. In this paper, GI is used as the backbone of the sequential pattern mining algorithm, which has achieved faster and higher performance accuracy than other sequential pattern mining algorithms for protein fold recognition.

In this paper, we introduce a Classified Sequential Pattern mining technique for Protein Fold Recognition (CSPF). CSPF consists of two main phases: 1) Sequential pattern mining and 2) fold recognition. It handles gap constraints, uses data parallelization, and performs incremental updating. CSPF has shown efficient results when applied to 36 SCOP protein folds. This paper is organized as follows: section 2 explains the proposed CSPF technique. Section 3 describes datasets used and the performance study. Finally, section 4 gives the conclusions and future work.

II. METHODS

CSPF technique consists of two main phases: the sequential pattern mining phase and the fold recognition phase. In the sequential pattern mining phase, Mix & Test algorithm is developed, which is used as a training phase. In the fold recognition phase, unknown protein folds are predicted via a proposed testing function. Our work is close to the sequential pattern mining suggested in [13]. However, this work depends on a new algorithm for sequential pattern mining, based on grammatical inference. In addition, it employs parallel sequential pattern mining and incremental updating.

A. Phase I: Sequential Pattern Mining:

During this phase, Mix & Test algorithm is developed in order to mine sequential patterns for each fold, based on Grammatical Inference. The key advantages of Mix &Test algorithm are minimizing I/O costs via one database scan, discovering combinations directly from sequences in-memory without searching the whole sequences file, no database projection, handling gaps, and working with variant length sequences without having to align them. In addition, Mix & Test algorithm supports incremental updating, where it does not prune infrequent patterns and count the support of them during the mining steps. Mix & Test algorithm acts iteratively. First, it generates a list of no gap sequential combinations, which will serve as the seed for the coming generation if there is a gap value specified. If no gap is specified, this list will be evaluated by the testing strategy with the specified minimum support threshold. Thus, this list will obtain the frequent and infrequent lists. If the gap value is specified, Mix &Test will loop to the combinations generation step and will use the combinations list obtained from the previous step to construct new combinations list with a gap by following steps of Mix &Test algorithm's grammar.

The steps of the algorithm are shown in Fig. 1.

1) Mix Strategy:

Problem Definition: Given a sequences file S that contains a set of sequences S= $\{s_1, s_2, ..., s_m\}$ and a set of items I = $\{ i_1, i_2, \ldots, i_n\}$ that may appear in any sequence (here, a set of amino acids), where m is the number of sequences in a file and n is the number of amino acids. A sequence $s_j = <i_1, ..., i_n>$, where i_1 is the first item in the sequence and i_n is the last item in the sequence. Let P is a subsequence that is derived from s_j, P_t is the current generated subsequence. P_{t-1} is the previous generated subsequence. The first generated subsequence will be:

$$P_1(s_j) = i_{n-1} \& i_n \quad (1)$$

The generated subsequence will be:

$$P_t(s_j) = i_{n-t} \& P_{t-1}(s_j) \quad (2)$$

1. Read New Protein Sequences
2. Apply Mix Strategy to generate sequential combination
3. If New Combination then
   Add new combination to Arraylist with support =1
   Else
   Increase it support by 1
4. If End of sequences file then
   If stopping criterion is reached (No_of Max gaps) then
   If Combinations' support >=Minsup then
   Output frequent Sequential patterns
   Else
   Output Infrequent Sequential patterns
   Else
   GOTO step 3
   Else
   GOTO step 2

Fig. 1. Mix & Test Algorithm Flowchart

Sequential combinations Generation "No-Gap combinations"

Mix strategy will first generate all "no gap combinations" list. It starts by reading the first sequence of protein sequences file and generates all possible sequential combinations of it. Mix strategy inserts the generated combination to the "no gap combinations" list with support equals to 1. Mix strategy will loop through new generated P to generate all possible combinations of it, using a removing procedure. This procedure removes the last item of the last generated combination to get a new combination from current P. It will stop generate P_t when t equals to number of items in the sequence n. An example of generated sequential combinations of "No-gap combinations" is illustrated in Table I, given original sequence MAKNNGCDP. After generating all possible sequential combinations from the first sequence of the protein sequences file. It will start to read the second sequence and go through the previous steps and generate all new combinations. If the new generated combination is previously composed, its support will be incremented by one; otherwise, it

will be inserted to "no gap combinations" list with support equals to 1, as clarified in Fig. 1.

Gapped Sequential combinations Generation

If there is a gap value specified, the "no gap combinations" list will be used to generate "one gap combinations" list, which will be used to generate "two gaps combinations" list, and so on. Mix strategy will use two procedures to generate all possible gapped sequential combinations: Ladder and CrissCross procedures.

First, the Ladder procedure reads each combination in "no gap combinations" list and loops through it by inserting one gap at a time starting from the second character position shifted right in each loop until reaching the last character of the combination. Then, it will start again to read the next no gap combination and apply the previous steps on it.

TABLE I. LIST OF GENERATED SEQUENTIAL COMBINATIONS "NO-GAP COMBINATIONS"

SUBSEQUENCE	LIST OF GENERATED COMBINATIONS
P1	DP
P2	CD, CDP
P3	GC, GCD, GCDP
P4	NG, NGC, NGCD, NGCDP
P5	NN, NNG, NNGC, NNGCD, NNGCDP
P6	KN, KNN, KNNG, KNNGC, KNNGCD, KNNGCDP
P7	AK, AKN, AKNN, AKNNG, AKNNGC, AKNNGCD, AKNNGCDP
P8	MA, MAK, MAKN, MAKNN, MAKNNG, MAKNNGC, MAKNNGCD, MAKNNGCDP

Definition 1: Given C as a "no gap combinations" list. C_i is a no gap combination. Let L be the gapped combination list generated by Ladder procedure, as follows:

$$L_y(C_i(S_j)) = C_{i\ -\ iy+1} \quad (3)$$

Where $L_y(C_i(S_j))$ is the y combination generated by Ladder procedure from no gap combination C_i, and i_{y+1} is the item i with the position y+1 in C_i combination.

Consider the first combination in the "No gap combinations" list is MAKNNGCDP, applying this procedure, we will obtain these one gap combinations: M_KNNGCDP, MA_NNGCDP, MAK_NGCDP, MAKN_GCDP, MAKNN_CDP, MAKNNG_DP, and MAKNNGC_P. Note that MAK_NGCDP is equivalent to MAKN_GCDP, so that they are treated as one combination and inserted only once in "one gap combinations" list as MAKNGCDP.

Second, the Crisscross procedure generates the rest of possible gapped sequential combinations of "one gap combinations" list. It reads each combination in "no gap combinations" list, looping through it and inserting one gap between each character of combination's characters. It starts from the second character's position shifted right one character position in each loop.

Definition 2: Given C as a "no gap combinations" list. C_i is a no gap combination. Let Q be the gapped combination list generated by Crisscross procedure, as follows:

$$Q_r(C_i(S_j)) = C_i - (i_{r+1}\ \&\ i_{r+3}\ \&\ i_{r+5}\ \&\ i_{r+7} \ldots i_n) \quad (4)$$

Where $Q_r(C_i(S_j))$ is the r combination generated by Crisscross procedure from no gap combination C_i, and i_{r+1} is the item i with the position r+1 in C_i combination. The concatenation part of the function will stop when n equals to or greater than the number of items in C_i.

By applying this procedure in the last example, MAKNNGCDP no-gap combination will produce: M_K_N_C_P, MA_N_G_D, MAK_N_C_P, MAKN_G_D, MAKNN_C_P, MAKNNG_D, and MAKNNGC_P. Notice that all these derivative combinations by the two procedures will take the same support of the parent no gap combination which they are derived from it. Mix strategy will stop generating new combinations when the number of sequences in protein sequences file. The final result from applying the Mix strategy will be a list of all combinations derived from all combinations lists.

2) Test strategy:

The Test strategy will filter final combinations list, which contains all no-gap and gapped combinations to distinguish frequent and infrequent patterns, according to user-specified support. However, infrequent patterns will not be discarded because incremental updating will be performed later on.

The most time consuming step in the Mix&Test algorithm is updating the combinations list, where a search is required in order to ensure if the generated combination is a new one to insert it or an old one to update its support. Thus, the combinations list may become very large. Therefore, a lexicographic prefix tree of lists is suggested, where each list contains all combinations with the same prefix. For example, let P = {p_1, p_2, … , p_n} be a set of lists (here n= 20 Amino Acids). Each p_i represents a list of all combinations with a prefix i. For example, if i = M, the list P_m can contain combinations, such as MV, MVV, MTV, MNKLSV. After Mix strategy generates the new combination, the first character of this combination is checked to determine which list to be inserted in. So, instead of having one big list, we will have p_n lists, this shrinks time T to find or insert combination to T/n. In order to increase the speed of computing and minimize the time required to generate the combinations in Mix strategy, especially with the large number of files and the rapid incoming rates, Parallel Mix strategy (PMix) is proposed. PMix uses horizontal data parallelization, where the data are split into chunks in the memory for the task. These data chunks will be distributed on PMix threads. Each thread will apply Mix strategy to generate the combinations of candidate patterns of this data chunk. After all threads finish their work, a combination integrator module will integrate all combinations generated from the threads into one final combinations list. The final combinations list is used by combinations evaluator module, which applies test strategy to get frequent and infrequent patterns.

3) Incremental updating

CSPF saves and records the sequential patterns of each fold, which are generated from the training phase. However, increasing the speed of processing, especially with large volumes of data and high data rates, is highly required. Existing incremental updating algorithms are highly based on the availability of main memory. As a result, the use of In-Memory relational databases is proposed, where TimesTen Oracle database management system is applied. TimesTen is an In-Memory DBMS technology, which provides very fast data access time because all its data will reside in physical memory (RAM) during run time. TimesTen provides applications with short, consistent response times and very high throughput required by applications with database-intensive workloads.

Incremental updating handles two cases: inserting new data and deleting old data. First, Insert module, as shown in Fig. 2, deals with new protein files to existing fold trial, the Mix strategy is applied to obtain the combination patterns of these files. These patterns are sent to database and added to the previously obtained frequent sequential patterns. Updated patterns can be classified into four cases: 1) Patterns that were frequent in the old database and become infrequent in the new database, 2) Patterns that were frequent in the old database and still frequent in the new database, 3) Patterns that were infrequent in the old database and become frequent in the new database, and 4) Patterns that were infrequent in the old database and still infrequent in the new database. Second, the Delete module deals with deleted sequences from the original database, which yields an inconsistent state with respect to the same specified minimum support threshold. The Delete procedure is similar to the Insert procedure. When deleting some protein sequences from existing fold trial, the obtained lists of frequent and infrequent patterns are affected. Delete module provides two ways for deletion either by deleting files directly by specifying their names or by a range of time to delete files in between.

B. Phase II: Protein Fold Recognition

The objective of the fold recognition phase is to classify unknown protein folds. In addition, an incremental updating module is used for maintaining the underlying database.

1) Weight Function for Protein Fold Recognition

The proposed weight function classifies the unknown protein by matching the extracted sequential patterns of each fold with the coming protein sequence. A weight for each fold with respect to the unknown protein is calculated. The higher the number of matched patterns is found, the higher the weight for the fold and the higher the probability of it to be selected as the recognized fold. However, there are very important aspects that have to be considered: 1) The length of the matched sequential patterns. The more matched frequent patterns with long length are reached, the higher the accuracy of the fold classification. 2) Two folds having the same number of sequential patterns. The proposed Weight Fold Function is:

$$W_f = N/S + \sum (K_i * (L_i / M_i)) \quad (5)$$

Where N is Number of matched Patterns, M is the Maximum length of extracted patterns for the fold, L is Length of pattern, K represents Number of patterns with the same length, S is the number of extracted sequential patterns for a fold, and W is the weight of the fold.

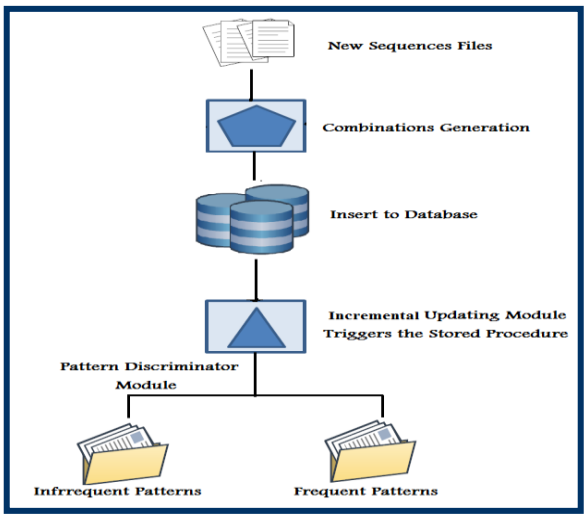

Fig. 2. Insert Module

III. Application

The CSPF technique is evaluated using different parameters, such as different support thresholds, number of sequences, memory consumption, and number of items per sequence. CSFP is trained and tested by a specific set of selected folds from the Structural Classification of Proteins (SCOP) database[1]. The ASTRAL SCOP 1.75B dataset updated on 25-4-2013 is selected, where no proteins with more than 40% identity between them are included. The ASTRAL SCOP 1.75B dataset release has 49,757 PDB entries and 136,776 Domains. For each fold in this set, a corresponding set of at least 30 protein members is obtained from Protein Data Bank (PDB) [44], which is a worldwide archive of structural data of biological macromolecules. The protein sequences extracted from this release are used to validate the results of the proposed model. Two third of this dataset is used in the training phase to establish features set for each fold and one third is used in the test data to check validity of the proposed model. The algorithms are developed using Java language with NetBeans IDE 7.2 as the Java execution environment. The algorithms are tested on an Intel Core™ i5 2.50 GHz with 6 GB of main memory. The operating system used is Windows 7.

The following performance evaluation tests are achieved: 1) For no gap mix strategy: a) Comparison of Mix & Test, PMix, and SPAM in terms of varied number of sequences, b) Comparison of Mix& Test, PMix, SPAM, and PrefixSpan in case of varied support threshold, and c) Comparison of Mix& Test, PMix, SPAM, and PrefixSpan in case of changing number of items per sequence. 2) For gapped mix strategy: Comparison of Mix & Test, and cSPADE algorithms according to the changes in maximum gap value. 3) Incremental Updating, 4) Memory consumption, and 5) Fold recognition phase: a comparison between the proposed method and SAM, which is widely used as a benchmark in fold

[1]http://scop.berkeley.edu/

recognition [39,45]. However, SAM requires higher computational effort during training, since it employs the Baum–Welch algorithm for training the model, which is an iterative procedure.

A. Performance analysis of no gap mix strategy

1) Number of sequences Test:

In this study, we measure the performance of Mix & Test, PMix, and SPAM algorithms according to the change in number of sequences. Fig. 3 shows the performance results derived from Mix &Test, PMix, and SPAM having data ranges from 100, 000 to 900,000 sequences. Fig. 4 illustrates the performance results derived from Mix&Test, PMix, and SPAM having data ranges from 1,000,000 to 5,000,000 sequences. In both figures, Mix &Test and PMix outperform SPAM, where time taken by them is much smaller than time taken by SPAM. In addition, PMix outperforms both Mix & Test and SPAM algorithms because of parallelization step.

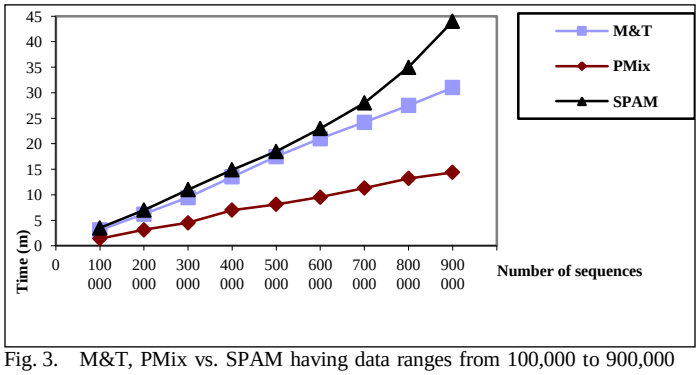

Fig. 3. M&T, PMix vs. SPAM having data ranges from 100,000 to 900,000 sequences

2) Minimum Support Threshold test:

Fig. 5 and Fig. 6 show the processing time of Mix&Test and PMix versus PrefixSpan and SPAM at different values of support threshold having the number of sequences equals 25,000 and 50,000, respectively. For protein sequences data and with very low minimum support threshold, the performance of PrefixSpan and SPAM take hours to process. On the other hand, Mix&Test and PMix take seconds and are not affected with the change of minimum support threshold values.

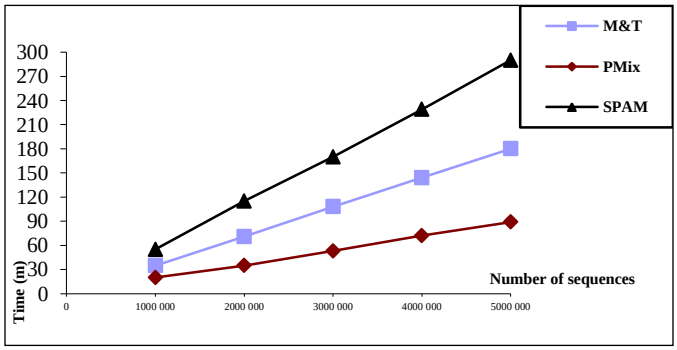

Fig. 4. M&T, PMix vs. SPAM having data ranges from 1,000,000 to 5,000,000 sequences

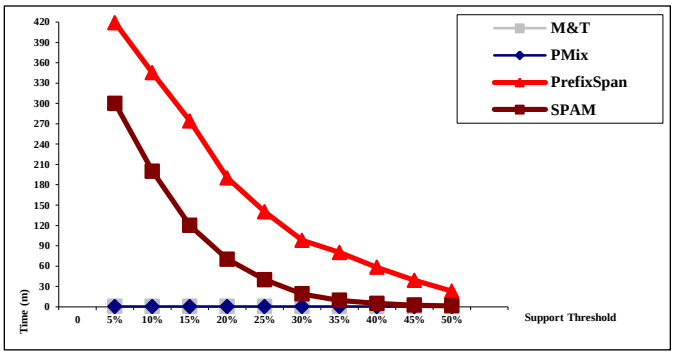

Fig. 5. Mix & Test, PMix, PrefixSpan, and SPAM Comparisons with varied support threshold (25,000 Sequences)

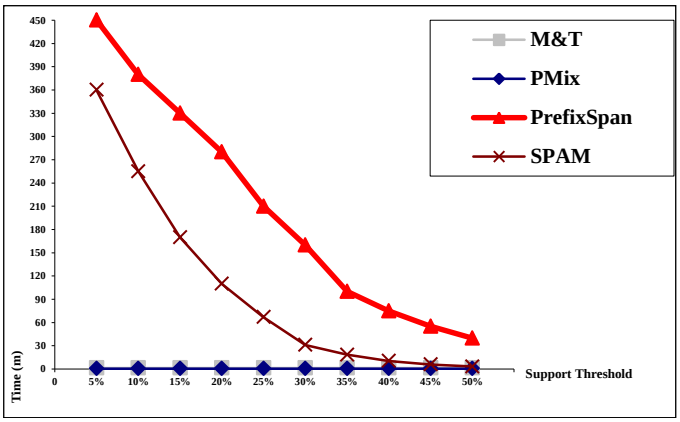

Fig. 6. Mix & Test, PMix, PrefixSpan, and SPAM Comparisons with varied support threshold (50,000 Sequences)

3) Number of Items per Sequence

Four tests are applied, having 180 and 300 items per sequence (ips) and variant support threshold, as shown in Fig. 7(a,b), respectively . Each trial in each test of the experiment is represented by adding 5% to the support threshold value of the previous trial. Thus, the first trial with support threshold value equals to 5% and the last one with support threshold value equals to 50%. The execution time is measured in each trial. The result of these tests shows the relationship between the value of the support threshold and the processing time in seconds according of the four algorithms: Mix& Test, PMix, PrefixSpan, and SPAM. As shown in Fig. 7(a,b), Mix & Test and PMix are much faster than PrefixSpan and SPAM.

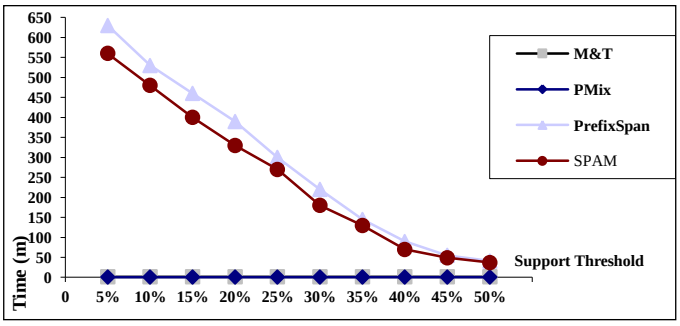

Fig. 7. (a). M&T and PMix vs. PrefixSpan and SPAM under different support threshold and 180 items per sequence

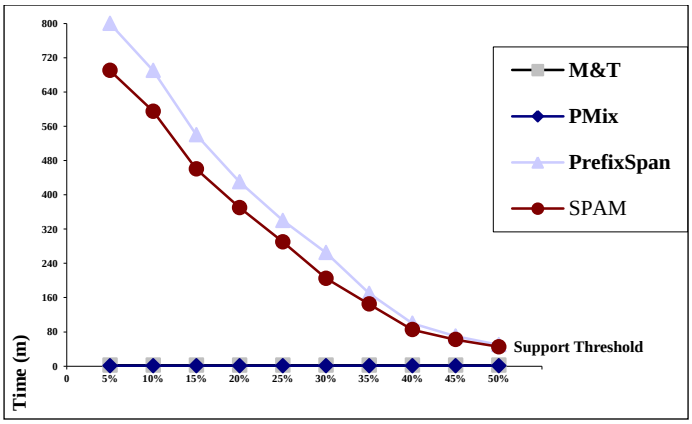

Fig. 7. (b).M&T and PMix vs. PrefixSpan and SPAM under different support threshold and 300 items per sequence

B. Performance analysis of gapped mix strategy

In this case, the performance of Mix&Test and PMix versus cSPADE algorithm is tested, according to the changes in maximum gap value, as illustrated in Fig. 8. This minimum support threshold equals to 35%. One can observe that the higher the gap value, the higher consumed time taken, having Mix&Test and PMix algorithms outperform cSPADE in small gap values. In addition, PMix outperforms both Mix&Test and cSPADE.

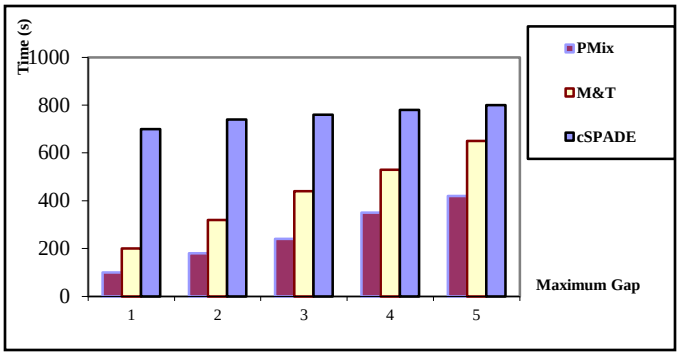

Fig. 8. Mix&Test and PMix vs. cSPADE under different Maximum Gap Values

C. Performance analysis of Incremental Updating Process

The Incremental updating module is implemented via two different database management systems. The first is MySQL DBMS with a conventional disk-resident database and the other is the Oracle TimesTen database, as explained previously. The performance of Mix&Test(TimesTen) and Mix&Test(MySql) according to the change in number of sequences (in this case from 10,000 to 50,000 sequences) is tested. In this case, a support threshold value equals to 20% with no gap value is applied, as illustrated in Fig. 11. In addition, the performance result of Mix&Test(TT) outperforms Mix&Test(MySql). Mix&Test(TT) takes around 30 seconds to process 10,000 sequences file where M&T(MySql) takes around 200 seconds to process it. This is because Timesten database is more efficient than MySql DBMS, where it offers a small, fast multithreaded, and transactional database engine with in-memory and disk-based tables.

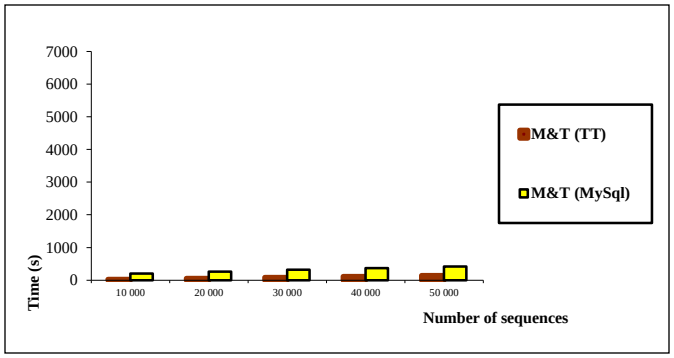

Fig. 9. Mix&Test(TT) and Mix&Test(MySql) under different Sequences File volumes

D. Performance Analysis of Memory Consumption

To evaluate the memory consumption of Mix&Test and PMix are evaluated versus cSPADE under two aspects, which are the different gap values and the variant number of sequences. Changing gap values, Mix& Test and PMix are tested versus cSPADE algorithm by using sequences file with 30,000 sequences with minimum support threshold value equals to 30%, as illustrated in Fig. 10. PMix consumes memory greater than Mix&Test because it processes multithreads in the same time. Also, cSPADE consumes much memory more than both Mix& Test and Pmix.

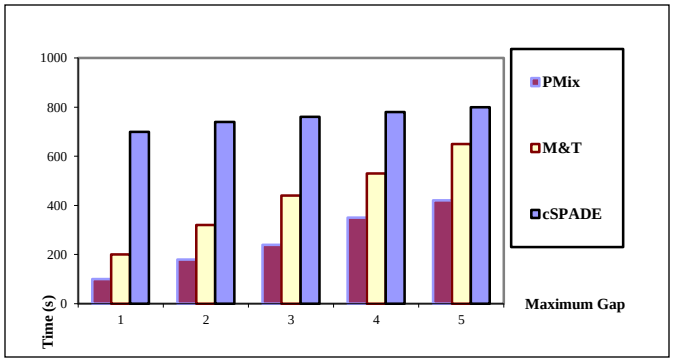

Fig. 10. The memory consumption of M&T and PMix vs. cSPADE under different gap values

E. Performance Analysis of Fold recognition Phase:

The fold recognition phase of CSPF technique is trained and tested by the dataset described previously [13]. In Table II, we compare the sensitivity of the CSPF to SPM sensitivity for fold recognition. Sensitivity of each model represents the number of proteins, which are classified successfully from the whole proteins under evaluation.

CSPF reported an overall accuracy of training data equals to 75.84%, with MaxGap=0 and MinSup=20%, while the overall accuracy of "SPM for FR" model is 59.7% with MaxGap=3 and MinSup=40%. A set of 804 protein experiments (test data set) are used to measure the accuracy of the model with the test set. CSPF reported an overall accuracy of testing data equals to 34.32%, as shown in Table III.

TABLE II. SENSITIVITY FOR ALL FOLDS AND OVERALL ACCURACY OF THE PROPOSED CSPF TECHNIQUE AND "SPM FOR FOLD RECOGNITION (FR)"

Fold index	CSPF Sensitivity		SPM for FR Sensitivity	
	(Proteins)	(%)	(Proteins)	(%)
a1	20/21	95.2	15/21	71.4
a3	20/20	100	17/20	85
a4	38/103	36.89	30/103	29.1
a24	28/28	100	28/28	100
a39	27/31	87.09	26/31	83
a60	21/25	84	19/25	76
a118	30/32	93.75	28/32	87.5
Class A (total)	184/260	70.76	163/260	62.7
b1	81/132	61.36	68/132	51.5
b2	20/20	100	19/20	95
b18	21/21	100	20/21	95.2
b29	22/24	91.6	21/24	87.5
b34	22/44	50	10/44	22.7
b40	26/61	42.6	25/61	41
b47	25/25	100	24/25	96
b55	18/24	75	16/24	66
b82	22/28	78.5	20/28	71.4
b121	27/27	100	26/27	96.3
Class B (total)	284/406	69.95	249/406	61.3
c1	82/143	57.34	16/143	11.2
c2	88/91	96.70	85/91	93.4
c3	20/22	90.9	22/22	100
c23	49/58	84.4	30/58	51.7
c26	31/35	88.57	29/35	82.9
c37	79/91	86.8	32/91	35.2
c47	34/39	87.1	22/39	56.4
c55	31/31	100	30/31	96.8
c56	18/20	90	20/20	100
c66	36/40	90	27/40	67.5
c67	30/31	96.77	31/31	100
c69	32/34	94.1	29/34	85.3
c94	22/23	95.6	19/23	82.6
Class C (total)	552/658	83.8	392/658	59.6
d15	39/44	88.6	21/44	47.7
d17	18/20	90	14/20	70
d58	38/102	37.25	22/102	21.6
d144	21/23	91.3	22/23	95.7
Class D (total)	116/189	90.4	79/189	41.8
f23	20/25	80	16/25	64
Class F (total)	20/25	80	16/25	64
g3	62/68	91.1	60/68	88.2
Class G (total)	62/68	91.1	60/68	88.2
Overall	1218/1606	**75.84**	959/1606	59.7

Using the same test datasets and in order to compare the efficiency of the proposed model, SAM model [16] is also employed. A comparison of the results obtained by CPSF, "SPM for FR" and SAM (E-values ranking) are presented in Table IV.

CSPF outperforms the other two models, where it reports an overall accuracy of testing data equals to 34.32% while the overall accuracy of "SPM for FR" model was 24.9% and SAM's overall accuracy was 29.4%. The classification results of the proposed method CSPF, and "SPM for FR" algorithm and SAM (E-values) of the test set are shown in Table IV.

In terms of space complexity, for a sequence file with n as the number of sequences, and m as the number of items per sequence and number of items equals to 20 which is the 20 amino acids, the space complexity of Mix&Test algorithm is O(20m+n). In terms of time complexity, the complexity of generating all the candidate patterns of Mix&Test with no gap is O(n2). The complexity of generating all the candidate patterns of Mix&Test with a gap m is O(n2)*m. The complexity of discovering the frequent patterns is O(N).

IV. CONCLUSIONS

In this work, we proposed a CSFP technique for protein fold recognition. This technique consisted of two main phases: sequential patterns extraction and protein fold recognition. Sequential patterns extraction phase introduced Mix & Test algorithm. Several experiments were conducted to assess the performance of Mix&Test and PMix. The performance of M&T and PMix algorithms were compared with PrefixSpan, SPAM and cSPADE algorithms.

In addition, performance of CSFP fold recognition was compared with "SPM for FR" and SAM (E-values) models. CSPF outperformed "SPM for FR" and SAM (E-values) models with an overall accuracy for training data equals to 75.84% and "SPM for FR" model was 59.7% for testing data. Future work of CSFP can be in several directions: utilizing optimization techniques to enhance the prediction results and applying high performance computing to provide very fast process over protein sequences databases. In addition, more protein sequences will be used.

TABLE III. DETAILED SENSITIVITY RESULTS FOR ALL FOLDS UNDER EVALUATION AND OVERALL ACCURACY OF THE PROPOSED CSPF MODEL IN THE TEST SET

Fold index	CSPF Sensitivity (Proteins)	CSPF Sensitivity %
a1	4/11	36.36
a3	8/10	80
a4	3/52	5.7
a24	15/15	100
a39	11/15	37.3
a60	2/12	16.3
a118	3/16	18.75
Class A (total)	46/131	35.11
b1	31/66	46.9
b2	2/10	20
b18	3/10	30
b29	2/12	16.6
b34	11/22	50
b40	12/31	38.7
b47	10/12	83.7
b55	2/12	16.6
b82	0	0
b121	9/14	64.3
Class B (total)	82/203	40.39
c1	2/71	2.8
c2	36/46	78.2
c3	2/11	18.1
c23	11/29	37.9
c26	7/17	41.1
c37	9/46	19.5
c47	1/20	5
c55	1/15	6.6
c56	0	0
c66	3/20	15
c67	8/15	53.3
c69	3/17	17.6
c94	9/12	75
Class C (total)	92/329	27.9
d15	7/22	31.8
d17	1/10	10
d58	8/51	15.6
d144	3/12	25
Class D (total)	19/95	20
f23	8/12	66.6
Class F (total)	8/12	66.6
g3	29/34	85.2
Class G (total)	29/34	85.2
Overall	276/804	34.32

TABLE IV. CLASSIFICATION RESULTS OF THE PROPOSED METHOD CSPF, "SPM FOR FR" ALGORITHM AND SAM (E-VALUES) IN THE TEST SET

Fold index	CSPF Sensitivity %	SAM(E-values) Sensitivity%	SPM for FR Sensitivity %
a1	36.36	81.8	18.2
a3	80	60	20
a4	5.7	3.8	28.8
a24	100	6.7	33.3
a39	37.3	87.7	66.7
a60	16.3	16.7	16.7
a118	18.75	0	37.5
Class A (total)	35.11	25.2	32.1
b1	46.9	50	36.4
b2	20	0	30
b18	30	30	20
b29	16.6	25	8.3
b34	50	36.4	0
b40	38.7	6.5	19.4
b47	83.7	83.3	58.3
b55	16.6	25	0
b82	0	14.3	0
b121	64.3	7.1	64.3
Class B (total)	40.39	32	25.6
c1	2.8	14.1	0
c2	78.2	23.9	69.6
c3	18.1	100	9.1
c23	37.9	27.6	24.1
c26	41.1	11.8	47.1
c37	19.5	80.4	10.9
c47	5	25	0
c55	6.6	13.3	0
c56	0	10	0
c66	15	20	5
c67	53.3	80	46.7
c69	17.6	5.9	5.9
c94	75	25	58.3
Class C (total)	27.9	32.5	21
d15	31.8	0	9.1
d17	10	0	0
d58	15.6	3.9	3.9
d144	25	91.7	16.7
Class D (total)	20	13.7	6.3
f23	66.6	25	41.7
Class F (total)	66.6	25	41.7
g3	85.2	44.1	76.5
Class G (total)	85.2	44.1	76.5
Overall	34.32	29.4	24.9

REFERENCES

[1] A. Sharaf Eldin, T. H.A. Soliman, M. E. Marie, and M. M. Ghareeb, "A Deep Glimpse into Protein Fold Recognition," International Journal of Sciences, vol. 2, pp.24-33, 2013.

[2] D. T. Jones, "GenTHREADER: An efficient and reliable protein fold recognition method for genomic sequences," J. Mol. Biol. vol. 287, pp.797-815, 1999.

[3] Lin K, May A and Taylor W (2002) Threading using neural network (TUNE): The measure of protein sequence-structure compatibility. Bioinformatics, vol. 18, 1350-1357.

[4] W. Thomas, C. Igel, and J. Gebert, "Protein fold class prediction using neural networks with tailored early-stopping," IEEE International Joint Conference on Neural Networks IJCNN, vol. 3, pp.1693 – 1697, 2004.

[5] A. Raval, Z. Ghahramani, and D. Wild, "A Bayesian network model for protein fold and remote homologue recognition," Bioinformatics, vol. 18: pp.788-801, 2002.

[6] J. Xu, "Fold recognition by predicted alignment accuracy," IEEE/ACM Trans. Comput. Biol. Bioinform, vol. 2 , pp.157-165, 2005.

[7] M. Judy and K. Ravichandran, "A solution to protein folding problem using a genetic algorithm with modified keep best reproduction strategy," Proceeding of IEEE Congress on Evolution Computation, Sep. 25-28, Singapore, pp. 4776-4780, 2007.

[8] R. Unger, "The genetic algorithm approach to protein structure prediction," Struct. Bond., vol. 110 , pp.153-175, 2004.

[9] S. Han, B. Lee, S. Yu, C. Jeong, S. Lee, and D. Kim, "Fold recognition by combining profile-profile alignment and support vector machine," Bioinformatics, vol. 21, pp.2667-2673, 2005.

[10] W. Chmielnicki and K. Stapor, "Protein fold recognition with combined SVM-RDA classifier," Proceedings of the HAIS, Part I, LNAI 2010, vol. 6076, pp.162-169, 2010.

[11] F. Liang and W. Wong, "Evolutionary Monte Carlo for protein folding simulations," J. Chemi. Phy., vol.115, pp.3374-3380, 2001.

[12] N. Alione, "Parallel evolution strategy on grids for the protein threading problem," J. Parallel Distributed Computing, vol. 66, pp.489-1502,2006.

[13] C. Carpio, S. Sasaki, L. Baranyi, and H. Okada, "A parallel hybrid GA for peptide 3D structure prediction," Proceedings of the Workshop on Genome Informatics, Universal Academy Press, Tokyo, 1995.

[14] R. Islam and A. Ngom, "Parallel evolution strategy for protein threading," Proceedings of the 25th International Conference on Chilean Computer Science Society, IEEE Computer Society, Washington, DC., USA, pp. 2347 – 2354, 2005.

[15] D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga, "Aligning multiple protein sequences by parallel hybrid genetic algorithm," Genome Inform, vol. 13, pp.123-132, 2002.

[16] S. Thomas and N. Amato, "Parallel protein folding with STAPL," Proceedings of the IEEE 18th International Parallel and Distributed Processing Symposium, Washington, DC., USA., doi: 10.1109/IPDPS.2004.1303204, 2004.

[17] R. Agrawal and R. Srikant, "Mining sequential patterns," Proceedings of the 1995 International Conference on Data Engineering, pp.3–14, 1995.

[18] A. Brazma, I. Johansen, J. Vilo, E. Ukkonen, "Pattern Discovery and Biosequences," Honavar, V G, Slutzki G (eds.) ICGI, LNCS (LNAI) 2000, vol. 1433, pp.257-270, Springer, Heidelberg, 2000.

[19] M. Ester, "A top-down method for mining most specific frequent patterns in biological sequence data," Proceedings of the 2004 SIAM International Conference on Data Mining (SDM '04), pp. 90–101, 2004.

[20] K. Wang, Y. Xu, and J. Yu, "Scalable sequential pattern mining for biological sequences," Proceedings of the 2004 ACM International Conference on Information and Knowledge Management (CIKM '04), pp. 178–187, 2004.

[21] Y. Xiong, J. He, and Y. Zhu, "TOPPER: An algorithm for mining top k patterns in biological sequences based on regularity measurement," Proceedings of IEEE Bioinformatics and Biomedicine Workshops (BIBMW), pp.283-288, 2004.

[22] L. Chen and W. Liu W, "An algorithm for mining frequent patterns in biological sequence," Proceedings of the IEEE 1st International Conference on Computational Advances in Bio and Medical Sciences (ICCABS), pp. 63-68, 2011.

[23] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M. Hsu, "FreeSpan: frequent pattern-projected sequential pattern mining," Proceedings of the ACM 2000 SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '00), pp. 355–359, 2000.

[24] M. Zaki, "SPADE: an efficient algorithm for mining frequent sequences," Machine Learning, vol. 42 (1/2), pp.31–60, 2001.

[25] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "Mining sequential patterns by pattern-growth: the PrefixSpan approach," IEEE Transactions on Knowledge and Data Engineering, vol. 16 (11), pp.1424–1440, 2004.

[26] X. Yan, J. Han, and R. Afshar, "CloSpan: mining closed sequential patterns in large datasets," Proceedings of the 2003 SIAM International Conference on Data Mining (SDM '03), pp. 166–177, 2003.

[27] P. Tzvetkov, X. Yan, and J. Han, "TSP: mining Top-K closed sequential patterns," Knowledge and Information Systems, vol. 7 (4), pp. 438–457, 2005.

[28] J. Wang, J. Han, and C. Li, "Frequent closed sequence mining without candidate maintenance," IEEE Transactions on Knowledge and Data Engineering, vol. 19 (8), pp.1042–1056, 2007.

[29] C. Luo and S. Chung, "Efficient mining of maximal sequential patterns using multiple samples," Proceedings of the 2005 SIAM International Conference on Data Mining (SDM '05), pp. 64–72, 2005.

[30] J. Pei, J. Han, and W. Wang, "Mining sequential patterns with constraints in large databases," Proceedings of the 2002 ACM International Conference on Information and Knowledge Management (CIKM '02), pp. 18–25, 2002.

[31] X. Ji, J. Bailey, and G. Dong, "Mining minimal distinguishing subsequence patterns with gap constraints," Knowledge and Information Systems, vol. 11 (3), pp.259–296, 2005.

[32] E. Chen, H. Cao, Q. Li, and T. Qian, "Efficient strategies for tough aggregate constraint-based sequential pattern mining," Information Sciences; 178 (6), pp.1498–1518, 2008.

[33] H. Kum, J. Pei, W. Wang, and D. Duncan, "ApproxMAP: approximate mining of consensus sequential patterns," Proceedings of the 2003 SIAM International Conference on Data Mining (SDM '03), pp. 311–315, 2003.

[34] C. Kum, J. Chang, and W. Wang, "Sequential pattern mining in multi-databases via multiple alignment," Data Mining and Knowledge Discovery, vol.12, pp.151–180, 2002.

[35] J. Yang, P. Yu, W. Wang, and J. Han, "Mining long sequential patterns in a noisy environment," Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data (SIGMOD '02), pp. 406–417, 2002.

[36] H. Cheng, X. Yan, and J. Han, "IncSpan: incremental mining of sequential patterns in large databases," Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04), pp. 527–532, 2004.

[37] M. Lin, S. Hsueh, and C. Chang, "Fast discovery of sequential patterns in large databases using effective time-indexing," Information Sciences, vol.178(22), pp.4228–4245, 2008.

[38] M. Zaki, "Sequence Mining in Categorical Domains: Incorporating Constraints," Proceedings of the in 9th Int'l Conference on Information and Knowledge Management, Washington, DC, 2000.

[39] T. Exarchos, C. Papaloukas, C. Lampros, and D. Fotiadis, "Mining sequential patterns for protein fold recognition," Journal of Biomedical Informatics, vol. 41: 165–179, 2008.

[40] U. Sakakibara, "Grammatical Inference in Bioinformatics," IEEE Transactions on Pattern Analysis and Machine Intelligence 27(7), pp. 1051-1062, 2005.

[41] P. Peris, D. Lopez, and M. Campos, "Igtm: an algorithm to predict transmembrane domains and toplogy in proteins," BMC-Bioinformatics 9, pp.367-378, 2008.

[42] P. Peris, D. Lopez, M. Campo, and J. Sempere, "Protein motif prediction by grammatical inference," In Sakakibara, Y., Y., Kobayashi, S., K., Nishino, T., Tomita, E. (eds) ICGI 2006. (LNCS (LNAI), vol. 4201,

pp.175-187. Springer, Heidelberg, 2008.

[43] T. Yokomori and S. Kobayashi, "Learning Local Languages and Their Application to DNA Sequence Analysis," IEEE Transactions Pattern Analysis Machine Intelligence, 20(10), pp.1067-1079, 1998.

[44] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, P. Bourne, "The Protein Data Bank," Nucleic Acids Research, Vol. 28, pp.235–242, 2000.

[45] D. Fischer, "Servers for protein structure prediction," Curr Opin Struct Biol, vol. 16(2), pp.178–182, 2006.