Iranian Journal of Operations Research
Vol. 3, No. 1, 2012, pp. 31-45

An Application of the ABS LX Algorithm to Multiple Sequence Alignment

S. Lalwani[1,*] , R. Kumar[2] , E. Spedicato[3] , N. Gupta[4]

We present an application of ABS algorithms for multiple sequence alignment (MSA). The Markov decision process (MDP) based model leads to a linear programming problem (LPP), whose solution is linked to a suggested alignment. The important features of our work include the facility of alignment of multiple sequences simultaneously and no limit for the length of the sequences. Our goal here is to avoid the excessive computing time, needed by dynamic programming based algorithms for alignment of a large number of sequences. In an attempt to demonstrate the integration of the ABS approach with complex mathematical frameworks, we apply the ABS implicit LX algorithm to elucidate the LPP, constructed with the assistance of MDP. The MDP applied for MSA is a pragmatic approach and entails a scope for future work. Programming is done in the MATLAB environment.

Keywords: *Multiple sequence alignment, ABS algorithm, Dynamic programming, Markov decision process.*

Manuscript received on 8/06/2011 and accepted for publication on 8/08/2011.

1. Introduction

Multiple sequence alignment consists of aligning three or more sequences of DNA, RNA, or protein at the same time. Multiple alignments arrange a set of sequences in a scheme, where positions believed to be homologous are written in a common column. MSA is of a central importance to bioinformatics and computational biology for the sequences identifying regions of similarity which may be a consequence of functional, structural, or evolutionary relationships. Although a large number of algorithms for computing MSA have been designed, the efficient computation of highly accurate multiple alignment is still a challenge [4].

The most widely used programs for global multiple sequence alignment are the Clustal series of programs [3]. ClustalW [19] incorporates a number of improvements to the alignment algorithm, and is based on a sensitive dynamic programming algorithm. ClustalW has given rise to a number of developments, including the ClustalX [18] which is the latest member of Clustal series. Although the alignments produced by ClustalX are the same as those produced by the latest release of ClustalW, the user can evaluate alignments better in ClustalX. ClustalW, version 1.8, is the most popular alignment program with excellent portability and operativity.

Other popular multiple sequence alignment methods are T-coffee [9] MUSCLE [5] and MAFFT [7]. The T-coffee method is broadly based on the popular progressive approach to multiple alignments. The data set of all pair wise alignments among the sequences is pre-processed using T-Coffee. Elements of the MUSCLE algorithm include fast distance estimation using *k*mer counting, progressive alignment using log-expectation score, and refinement using tree dependent restricted partitioning. MUSCLE uses two distance measures for a pair of sequences: a *k*mer distance (for an

*Corresponding Author.
[1] Birla Institute of Scientific Research, Jaipur-302001, India. Email: slalwani.math@gmail.com
[2] Department of Electrical Engineering, Malaviya National Institute of Technology, Jaipur, India. Email: rkumar.ee@gmail.com
[3] Department of Operation Research, University of Bergamo, Italy. Email: emilio@unibg.it
[4] Department of Mathematics, Malaviya National Institute of Technology, Jaipur, India. Email: n1_gupt@yahoo.com

unaligned pair) and the Kimura distance (for an aligned pair). MAFFT method for multiple sequence alignment is based on the fast Fourier transform (FFT), which allows rapid detection of homologous segments in pairs of sequences.

To the best of our knowledge, all the popular methods including ClustalW are based on constructing pairwise sequence alignment first. Then, as the classical dynamic programming algorithms perform pairwise sequence alignments, the results are combined to find multiple sequence alignment [6]. Unfortunately, this approach is impractical for alignments of more than a few sequences, due to its high computing cost. Therefore, several heuristics have been proposed to compute nearly optimal alignments, based on minimization of the total cost, so that large sequences may be aligned effectively.

The method of using an MDP approach for LPP is an efficient approach, as the multiple sequences can be aligned simultaneously, saving time and space for aligning a large number of long sequences [6]. We applied the implicit LX method [14] of ABS class of algorithms [1] to solve the LPP, constructed with the assistance of the MDP. Several computational experiments have confirmed that ABS methods can be implemented in a very stable way. Moreover, on vector/parallel machines, they are usually faster than the standard methods [2]. The computational results show that the best ABS methods are competitive. Additionally, ABS methods are reliable on ill conditioned and rank deficient problems.

We implemented the method for a family of human protein kinase sequences. Protein kinase is a key regulator of cell function constituting one of the largest and most functionally diverse gene families. Kinase human sequences were downloaded in FASTA format from: *http://www.uniprot.org*. We aimed to achieve statistically significant accuracy improvements comparable to one of the existing top performing aligner ClustalW2 from: *www.ebi.ac.uk/ClustalW2/*.

The remainder of our work is organized in 5 sections. Section 2 reviews some necessary concepts of ABS algorithms, the Implicit LX method and MDP. After defining the formulation of LPP using MDP in Section 3, we present the use of the ABS algorithm for MSA in Section 4. Section 5 presents our results obtained from the algorithm, verified with those produced by ClustalW2. We conclude in Section 6.

2. Definitions and Basic Concepts

Here, we give some necessary definitions and concepts given by Abaffy and Spedicato [1], Puterman [10] Sharma [12] and Ross [11].

2.1. ABS Algorithm

The ABS algorithm was introduced in 1984 by Abaffy, Broyden and Spedicato [1] to solve determined or underdetermined linear systems, and later extended to linear least squares, nonlinear equations, optimization problems and Diophantine equations (see [16] and [17]). The latest works deal with stochastic, infinite and fuzzy systems. The class of ABS methods unifies most existing methods for solving linear systems and provides a variety of alternative ways of implementing a specific algorithm [1, 15, 16, 17]. These methods result from collaboration of a number of mathematicians from most of Europe including Italy, Hungry and UK, as well as from China and Iran. Here, we use the ABS implicit LX method for solving LPP and some new applications as well. The steps of the basic or unscaled ABS class of algorithms for solving a linear system of the

form $Ax = b, A = (a_1, ..., a_m)^T \in R^{m \times n}$, are:

(a) *Set* $x_1 \in R^n$, *arbitrary*, $H_1 \in R^{n \times n}$, *arbitrary nonsingular*, $i = 1$.

(b) *Compute the vector* $s_i = H_i a_i$, *and the scalar* $t_i = a_i^T x_i - b_i$.

If $s_i \neq 0$ *then go to (c), the ith equation is linearly independent from the previous equations, else if* t_i*=0 then remove equation i, which is dependent on previous equations, set* $x_{i+1} = x_i$, $H_{i+1} = H_i$ *and go to (f), else stop (the system is inconsistent and lacks solution).*

(c) *Compute the search direction* $p_i = H_i^T z_i$, $z_i \in R^n$, *arbitrary save that* $z_i H_i a_i \neq 0$.

(d) *Update the estimation of the solution by* $x_{i+1} = x_i - \alpha_i p_i$, *with* $\alpha_i = \dfrac{t_i}{a_i^T p_i}$.

(e) *Update the Abaffian matrix* H_i *by* $H_{i+1} = H_i - \dfrac{H_i a_i w_i^T H_i}{w_i^T H_i a_i}$,

where $w_i \in R^n$ *is arbitrary save that* $w_i^T H_i a_i \neq 0$.

(f) *If* $i = m$ *then stop,* x_{m+1} *solves the system, else let i=i+1 and go to (b).*

Remarks: The Huang or implicit Gram-Schmidt algorithm is defined by the choices $H_1 = I, z_i = w_i = a_i$. The ABS algorithm determines the unique solution of least Euclidean norm of an underdetermined system, if it is started with an arbitrary vector x_1 proportional to a_1, usually the zero vector. If $x_1 = 0$, then the solution is moreover approached monotonically in norm from below. Additionally, the search directions p_i are orthogonal. The algorithm's stability can be improved in several ways. Usually one does a reprojection on the search direction, i.e.,

$$p_i = H_i(H_i a_i). \tag{1}$$

and then defines the update as

$$H_{i+1} = H_i - \frac{p_i p_i^T}{p_i^T p_i}. \tag{2}$$

This modification is called the modified Huang algorithm.

We use here the implicit LX algorithm [14], the best suited algorithm for solving LPP.

2.2. Implicit LX Method

For the implicit LX algorithm, we need to define the choice of the parameters in steps (a) and (b) of the ABS algorithm as follows:

$$H_1 = I,\ z_i = e_{k_i},\ w_i = \frac{e_{k_i}}{e_{k_i}^T H_i a_i}, \tag{3}$$

where, k_i is an integer, $1 \leq k_i \leq n$, such that

$$e_{k_i}^T H_i a_i \neq 0. \tag{4}$$

We assume that *A* has full row rank. There is at least one index k_i such that (4) is satisfied. For stability reasons, k_i is selected such that $\eta_i = \left| e_{k_i}{}^T H_i a_i \right|$ is maximized.

The implicit LX algorithm has the same operational cost as the Gaussian elimination but it does not require pivoting and has less intrinsic storage cost. Thus, it can be regarded as the computationally best known algorithm of the standard type for solving a general system [14, 17]. The implicit LX method has a natural application to LPP [13], say

$$\begin{aligned} &\min \quad c^T x \\ &\text{s.t.} \quad Ax = b \\ &\qquad x \geq 0. \end{aligned} \tag{5}$$

In the simplex method, a column of an original index N^* from the matrix A_N is interchanged with a column of an original index B^* in the matrix A_B. The column in A_N is often taken as the column with a minimal relative cost. In terms of the ABS formulation, this is equivalent to minimizing, with respect to $i \in N_m$, the scalar η_i, defined by

$$\eta_i = c^T H^T e_i . \tag{6}$$

The column in A_B to be exchanged is usually chosen with the criterion of the maximum displacement along an edge such that the basic variables remain non-negative. The scalar w_i is defined by

$$w_i = x^T e_i / e_i^T H^T e_{N^*}, \tag{7}$$

where, x is the current basic feasible solution. Then, the above criterion is equivalent to minimizing *w*, with respect to the set of indices $i \in B_M$, such that

$$e_i^T H^T e_{N^*} > 0. \tag{8}$$

The H-matrix is then updated by

$$H' = H - \left(He_{B^*} - e_{B^*} \right) e_{N^*}^T H / e_{N^*}^T He_{B^*}. \tag{9}$$

Equation (9) is an important relation in itself as well as for its application to the simplex method.

Finally, we refer to some numerical experiments on the implicit LX algorithm. ABS algorithms have been tested extensively on sequential and vector-parallel machines (Alliant, IBM 3090) versus codes from the NAG, LINPACK, LAPACK and MATLAB packages. The ABS codes were written in FORTRAN 77, except those produced for running on CERFACS Alliant FX80 with 8 processors, where a CERFACS version of FORTRAN 90 was used. The implicit LX method was tested on a digital workstation on 21 families of problems with the size of *n* up to 800, entries being exactly represented and treated as full problems [8, 17]. It was shown that the implicit LX algorithm is more accurate than the implicit LU, the modified Huang and the LAPACK LU solver,

specially for the relative solution error defined by $\eta = \left\| \tilde{x} - x^{+} \right\| / \left\| b \right\|$. The superiority is more evident on ill conditioned problems. Over all the problems, the implicit LX algorithm in double precision was better in 41 problems, MATLAB was better in 14 problems and accuracy differences were less than one percent in 5 problems. In single precision, the implicit LX algorithm was better in 25 problems, LAPACK was better in 17 problems and accuracies were less than one percent in 18 problems.

The method used here for MSA has shown to be better than the other classical approaches, as all the long multiple sequences are aligned at the same time. The problem was obtained using MDP.

2.3. Markov Decision Process

MDP, also known as stochastic dynamic programming, named after Andrey Markov, provides a mathematical framework for modeling decision-making, in situations where events are partly random and partly under the control of a decision maker. MDP is an extension of Markov chain; the difference is in the addition of actions (allowing choice) and rewards (giving motivation). Conversely, while ignoring rewards, if there were only one action (or if the action to take were fixed for each state), an MDP would reduce to a Markov chain.

MDPs are practiced to study a gamut of optimization problems solved via dynamic programming and reinforcement learning. MDPs constitute a basic framework for dynamically controlling systems that evolve in a stochastic way [10]. The evolution of the system is as shown in Fig. 1. The formal definition can be given by a 4-tuple set (S, A, P (-,-), R (-,-), as presented in Table 1.

Table 1: A Markov decision process

S	A finite set of states
A	A finite set of actions
$P_a(s, s')$	Probability that action a in state s at time t will lead to state s' at time $t + 1$
$R_a(s, s')$	The immediate reward received after transition to state s' from state s with transition probability $P_a(s, s')$

Let s_t denote the state at time $t \in \{0,1,...\}$ and a_t denote the action chosen at that time. If $s_t = s \in S$ and $a_t = a \in A$, then the system transitions from state s to state $s_{t+1} = s' \in S$ with probability $P_a(s,s')$, and a reward of $R_a(s,s')$ is obtained. The dynamic Bayesian network is shown in Fig. 2. Once the transition to the next state has occurred, a new action is chosen, and the process is repeated. The conditional probabilities defined in the process are:

Transition probability: $$P(s_{t+1} \mid a_t, s_t). \tag{10}$$

Reward probability: $$P(r_t \mid a_t, s_t). \tag{11}$$

Policy: $$P(a_t \mid s_t) = \pi(a_t \mid s_t). \tag{12}$$

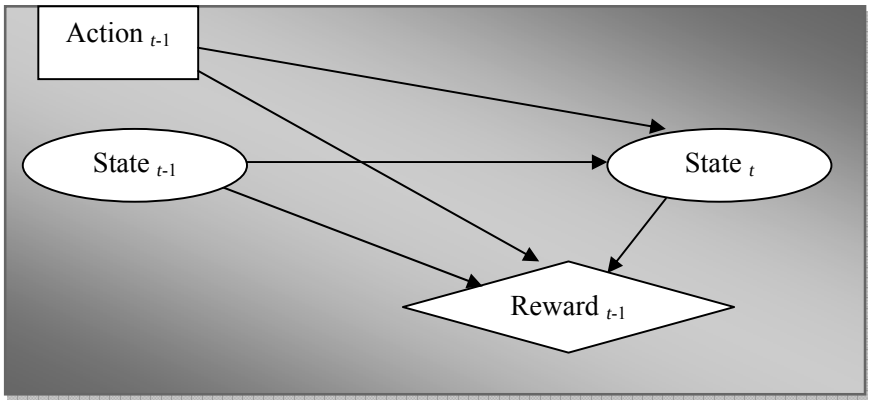

Figure 1: An MDP "standard" model

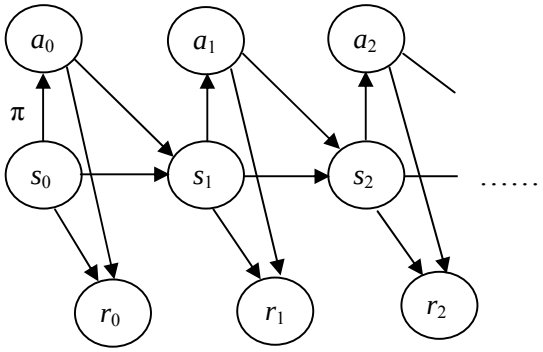

Figure 2: A dynamic Bayesian network for MDP

We obtained the solution of the MDP by formulating the problem in the canonical form of LPP, where the states, actions and corresponding rewards were used to formulate the constraints and the objective function of the LPP. The mathematical formulation of the problem is, as described below.

3. MDP Employed for LPP

The algorithm adduced in our work is supported by MDP, which constitutes LPP by articulating the probability matrix. LPP can be expressed in a canonical form as

$$\text{Maximize} \quad c^T x \tag{13}$$

$$\text{s.t.} \quad Ax \le b \ . \tag{14}$$

Equations (14) are the constraints specifying a convex polytope, over which the objective function is to be optimized.

The frequency of transition from one state to another for a particular action was computed using the following formula:

$$P_{ij}(a) = \frac{n(i,j,a)}{n(i,a)}. \tag{15}$$

After having the transitional probability matrix and scores for each sequence, we formulated the linear function as

$$Maximize \sum_j \sum_a y_{ja} R(j,a) \tag{16}$$

with the constraints for the given LPP as

$$\sum_j \sum_a y_{ja} = \frac{1}{1-\alpha}, \tag{17}$$

$$\sum_j y_{ja} = b_j + \alpha \sum_j \sum_a y_{ja} P_{ij}(a), \; y_{ja} \geq 0, \forall \, j,a, \tag{18}$$

where, a denotes the action, i represents the initial state, j is defined as the final state, y_{ja} means how frequently action a is opted from state j when the maximum total score is b_j, and b_j is the initial distribution of states, which is set to $1/m$, with m being the number of states in the sequence. The steps for forming LPP using MDP are:

Step 1: *Assign states to every column of multiple sequences.*

Step 2: *Assign actions for each state of all the sequences.*

Step 3: *Compute the transitional probability matrix for the state against the action.*

Step 4: *Compute the reward by using the BLOSUM62 matrix for every state and action.*

Step 5: *Formulate the objective function and the corresponding constraints.*

Hence, after formulating the problem as a linear program, the solution is obtained by applying the implicit LX method of the ABS algorithms.

4. Using an ABS Algorithm for MSA

The ABS methods has been broadly used for solving linear and nonlinear systems of equations comprising large number of constraints and variables, thereby saving time and effectively dealing with resulting complexities. Computational results are presented here using programs developed in MATLAB environment. The key challenges experienced by us were:

- Creating the transitional probability matrix.
- Calculating rewards using the Blosum62 matrix.
- Formulating the LPP.

The algorithm used in this work is endorsed by MDP which constitutes LPP by articulating the probability matrix. We targeted on aligning four sequences, so that there were 16 different possibilities of combining gaps and letters. These possibilities are represented in the form of '0' and '1'. A gap is represented by '0' and a letter by '1' (Table 2). Also, each column of the aligned sequence represents a state *s*, which is defined by a number. For example, if testing is done on 4 sequences, each of length 17, i.e.,

-	M	H	I	P	V	E	P	P	T	T	R	R	F	T	P	P
-	M	N	E	P	Q	S	G	P	D	F	S	K	Y	I	L	D
-	M	T	D	Q	E	S	L	I	E	S	F	T	K	R	I	A
-	M	S	E	K	Q	S	V	K	Q	Y	I	Q	G	K	L	D

then each column of the aligned sequence represents a state s, which is defined by a number as shown in Table 3.

Table 2: Representing actions

	0	0	0	0	1	0	0	1	0	1	1	0	1	1	1	1
	0	0	0	1	0	0	1	0	1	0	1	1	0	1	1	1
	0	0	1	0	0	1	0	0	1	1	0	1	1	0	1	1
	0	1	0	0	0	1	1	1	0	0	0	1	1	1	0	1
Action '*a*'	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**

Table 3: Representing states and actions

	-	M	H	I	P	V	E	P	P	T	T	R	R	F	T	P	P
	-	M	N	E	P	Q	S	G	P	D	F	S	K	Y	I	L	D
	-	M	T	D	Q	E	S	L	I	E	S	F	T	K	R	I	A
	-	M	S	E	K	Q	S	V	K	Q	Y	I	Q	G	K	L	D
State '*s*'	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Action '*a*'	**1**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**

In Table 4, '*a*' denotes the action, *i* represents the initial state, and *j* is defined as the final state. As a citation for the above example, the algorithm creates a matrix as shown in the table. These frequencies are stored in a matrix known as the transitional probability matrix, which is further used in solving the formed LPP. The score for every next upcoming state is calculated using the Blosum62 matrix [12], which is represented by $R(j,a)$. Transition from state j to another state with action a is represented by $R(j,a)$. For calculating this score, a pair of sequences is selected from the data set and then the corresponding Blosumn62 score is selected for that pair. Corresponding to the action, if there appeared a gap, a deletion penalty, *d*, is deducted, and if an extension with gaps is encountered, an extension penalty, e, is deducted, as exemplified in Fig. 3.

Table 4: Transitional probability matrix

i	j	P_{ij}	$P_{ij}(a)$	a
1	2	1	1	16
2	3	1	1	16
3	4	1	1	16
4	5	1	1	16
5	6	1	1	16
6	7	1	1	16
7	8	1	1	16
8	9	1	1	16
9	10	1	1	16
10	11	1	1	16
11	12	1	1	16
12	13	1	1	16
13	14	1	1	16
14	15	1	1	16
15	16	1	1	16
16	17	1	1	16

$$R\left(\begin{pmatrix}-\\-\\-\\M\end{pmatrix},\begin{pmatrix}0\\0\\0\\0\end{pmatrix}\right)=\begin{bmatrix}Bl(-,-)-e-e+Bl(-,M)-e-d+Bl(-,M)-e-d\\+Bl(-,-)-e-e+Bl(M,-)-e-d++Bl(-,-)-e-e\end{bmatrix}\Big/4$$

$$R\left(\begin{pmatrix}H\\N\\T\\S\end{pmatrix},\begin{pmatrix}1\\1\\1\\1\end{pmatrix}\right)=\begin{bmatrix}Bl(H,N)+Bl(H,T)+Bl(N,T)\\+Bl(T,S)+Bl(H,S)+Bl(N,S)\end{bmatrix}\Big/4$$

Figure 3: Reward for next upcoming state

As discussed, b_j is the initial distribution of states, which is set to $1/m$, where, m is the number of states in the sequence, taken to be 17 for the matrix of Table 2. Hence, Fig. 4 shows the initial distribution for the cited example.

$$b_j = \begin{bmatrix} 0.0588 \\ 0.0588 \\ . \\ . \\ . \\ 0.0588 \end{bmatrix}$$

Figure 4: Initial distributions b_j

The deletion penalty and the extension with gap penalty are defined to be $d = 4$ and $e = 2$, respectively, and the range for α is defined as $0 \le \alpha \le 0.9$. We set $\alpha = 0.5$.

Finally, we obtained the LPP as shown in the form of equations (16)-(18), constructed using MDP, and computed by the ABS LX method to incur the optimal and numerically stable results, with the aid of MATLAB.

5. Results

The proposed algorithm was tested on the data set of human protein kinase sequences taken in Fasta format (Link: www.uniprot.org/uniprot/, the p45 sequences). The results obtained from the algorithm were compared with those produced by ClustalW2; our results were consistent with those of ClustalW2. We randomly selected 4 sequences from the Fasta file and checked the execution time and accuracy of the obtained results.

Table 5: Comparison of execution time, sensitivity, PPV and *F*-measure

Length of the largest sequence (L)		L<200	200<L<300	300<L<400	400<L<500
Sensitivity	Proposed Algorithm	1	0.955	1	1
PPV	Proposed Algorithm	0.889	0.955	1	1
F-measure	Proposed Algorithm	0.941	0.955	1	1
Time (Seconds)	ClustalW2	31.04	27.67	37.84	41.01
	Proposed Algorithm	33.18	31.04	38.01	42.67

Table 5 presents the comparison of the execution time, sensitivity, PPV and *F*-measure between ClustalW2 and our proposed algorithm. The alignment is performed 120 times (30 times for each group of length) and the averages of the obtained values were noted. The result obtained by ClustalW was assumed to be correct, since the protein sequences were not experimentally validated. Hence, the sensitivity, PPV and F measure for ClustalW were assumed to be 1.

The results show that our proposed algorithm is comparable with ClustalW2. Thus, the proposed algorithm is considered to be a viable new approach for aligning multiple sequences simultaneously.

For sequences of length 407, the number of equations as the constraints for the LPP was 407 and the number of variables was 6, 512.

The sequence size being so large, the screenshots of the actual sequence cannot be viewed in a single window. Thus, the results obtained by the algorithm for the matrix of Table 2 are shown using the screenshots of the ClustalW2 and MATLAB windows (figures 5, 6 and 7). The solution for LPP is:

$$y_{2,16} = y_{3,16} = y_{4,16} = y_{5,16} = y_{6,16} = y_{7,16} = y_{8,3} = y_{9,16} = y_{10,16} = y_{11,16}$$
$$= y_{12,16} = y_{13,16} = y_{14,16} = y_{15,16} = y_{16,16} = y_{17,16} = y_{18,14} = 0.0588.$$

This represents the states and actions regarding the positions of the aligned amino acids. The remaining variables are all zero and represent no alignment.

An interpretation for the obtained results can be given as follows: for the variable $y_{2,16}$ it means that for state 2, action 16 is defined, and all nucleotides are aligned without any gap. Similarly, $y_{8,3}$ is meant for state 8, with action 3 taken, and inserting gap in all sequences except the third one. Hence, the optimal alignment is:

M	H	I	P	V	E	-	P	P	T	T	R	R	F	T	P	P
M	N	E	P	Q	S	-	G	P	D	F	S	K	Y	I	L	D
M	T	D	Q	E	S	L	I	E	S	F	T	K	R	I	A	-
M	S	E	K	Q	S	-	V	K	Q	Y	I	Q	G	K	L	D

After shuffling corresponding to ClustalW2 format, alignment can be written as follows:

M	N	E	P	Q	S	-	G	P	D	F	S	K	Y	I	L	D
M	S	E	K	Q	S	-	V	K	Q	Y	I	Q	G	K	L	D
M	T	D	Q	E	S	L	I	E	S	F	T	K	R	I	A	-
M	H	I	P	V	E	-	P	P	T	T	R	R	F	T	P	P

In this format, the results are the same, and only the order of the sequences has changed.

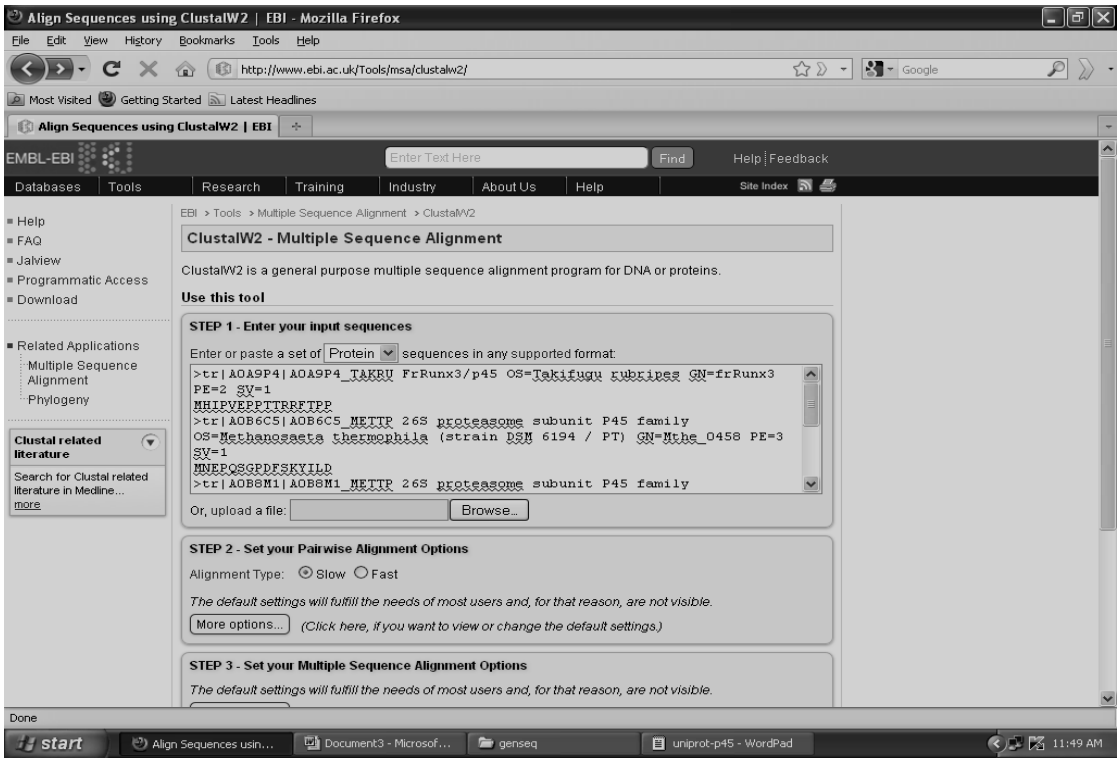

Figure 5: Screenshot of inserting the sequence in ClustalW2

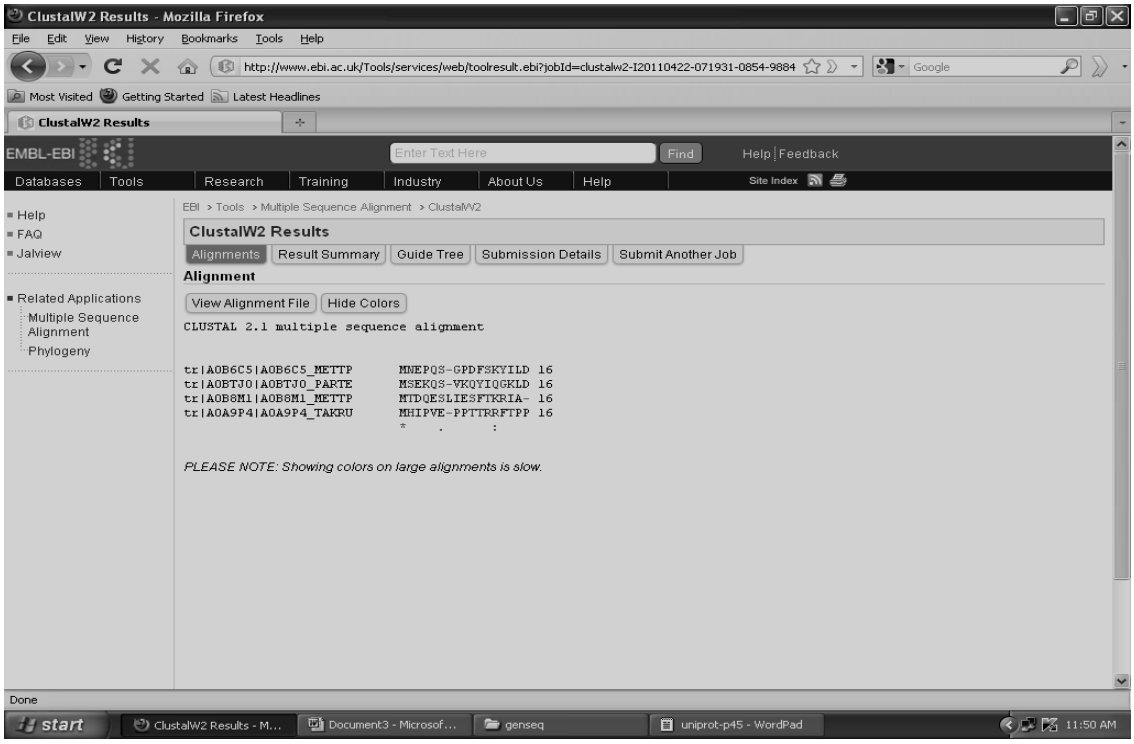

Figure 6: Screenshot of obtained alignment from ClustalW2

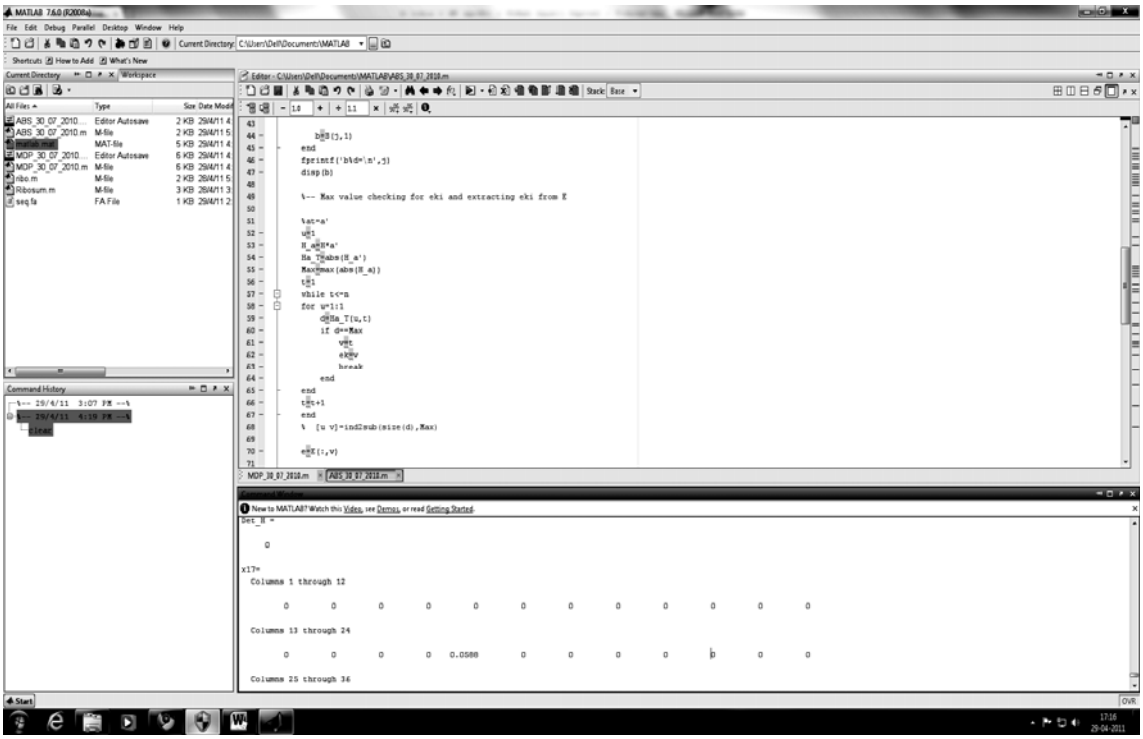

Figure 7: Screenshot of output using MATLAB

6. Conclusions

We devised a flexible software platform, allowing the user to align sequences, with cost functions and trained data of their choice and using the ABS implicit LX algorithm. In the literature, ABS algorithms have shown to be numerically more stable and efficient than many classical algorithms, especially when performed on parallel computers. The MDP approach for LPP is efficient to perform the simultaneous alignment of multiple sequences. The state of the art methods create pairs of sequences and then merge the results, and hence require excessive time and space. In our approach, the length of the sequence has no bar and there is no need to take pairs of sequences while aligning. The program aligns all the input sequences at a time. The user is allowed to give a long sequence; e.g., if the number of sequences is 4, then the number of actions will always be 16 and the number of states will have an upper limit of a total of 21 amino acids. The computing results showed our proposed approach to be effective in reducing expense encountered by dynamic programming based algorithms in aligning large numbers of sequences.

Acknowledgements

We gratefully acknowledge the kind support of Prof. Ghosh, Executive Director, Birla Institute of Scientific Research and Dr. Krishna Mohan, Head, R & D, Birla Institute of Scientific Research. We are thankful to BTIS-sub DIC, supported by DBT (Govt. of India) and Advanced Bioinformatics Centre, supported by Govt. of Rajasthan at Birla Institute of Scientific Research for providing us the infrastructure facilities for the present work. We acknowledge Priyanka Choudhary, six month project trainee at Birla Institute of Scientific Research, for her assistance in programming. We are particularly thankful to Nezam Mahdavi-Amiri, the Editor-in-Chief of IJOR for full reading of the manuscript and suggestions for improving the presentation.

References

[1] Abaffy, J. and Spedicato, E. (1989), ABS Projection Algorithms: Mathematical Techniques for Linear and Nonlinear Equations, Ellis Horwood Ltd., Chichester, England.

[2] Bertocchi, M. (1989), Numerical experiments with ABS algorithms for linear systems on a parallel machine, *Journal of Optimization Theory and Applications*, 60(3), 375-392.

[3] Chenna, R., Sugawara, H., Koike, T., Lopez, R., Gibson, T. J., Higgins, D. G. and Thompson, J. D. (2003), Multiple sequence alignment with the Clustal series of programs, *Nucleic Acids Research*, 31(13), 3497-3500.

[4] Durbin, R., Eddy, S., Krogh, A. and Mitchison, G. (1998), Biological Sequence Analysis, Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, United Kingdom.

[5] Edgar, R. C. (2004), MUSCLE: a multiple sequence alignment method with reduced time and space complexity, *Nucleic Acids Research*, 32(5), 1792-1797.

[6] Hunt, F.Y., Kearsley, A. J. and O'Gallagher, A. (2003), A linear programming based approach for multiple sequence alignment, *Proceedings of the Computational Systems Bioinformatics* (CSB'03), IEEE, 532-533.

[7] Katoh, K., Misawa, K., Kuma, K. and Miyata, T. (2002), MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform, *Nucleic Acids Research*, 30(14), 3059-3066.

[8] Mirnia, K. (1996), Iterative refinement in ABS method, Report DMSIA 32/96, University of Bergamo.

[9] Notredame, C., Higgins, D. G. and Heringa, J. (2000), T-coffee: a novel method for fast and accurate multiple sequence alignment, *Journal of Molecular Biology*, 302(1), 205-217.

[10] Puterman, M. L. (2005), *Markov* Decision Processes: Discrete Stochastic Dynamic Programming, Wiley-Interscience, John Wiley & Sons.

[11] Ross, S. M. (2007), Introduction to Probability Models, 9th edition, Academic Press, Elsevier.

[12] Sharma, K. R. (2008), Bioinformatics Sequence Alignment and Markov Models, McGraw-Hill Professional Publishing.

[13] Spedicato, E., Xia, Z. and Zhang, L. (1995), Reformulation of the simplex algorithm, Preprint, University of Bergamo.

[14] Spedicato E., Xia, Z. and Zhang, L. (1997), The implicit LX method of the ABS class, *Optimization Methods and Software*, 8(2), 99-110.

[15] Spedicato, E., Xia, Z. and Zhang, L. (2000), ABS algorithms for linear equations and optimization, *Journal of Computational and Applied Mathematics*, 124(1-2), 155-170.

[16] Spedicato, E., Bodon, E., Popolo, A.D. and Mahdavi-Amiri, N. (2003), ABS methods and ABSPACK for linear systems and optimization: A review, *4OR*, 1(1), 51-66.

[17] Spedicato, E., Bodon, E., Xia, Z. and Mahdavi-Amiri, N. (2010), ABS Method for continuous and integer linear equations and optimization, *Central European Journal of Operations Research*, 18(1), 73-95.

[18] Thompson, J. D., Gibson, T. J., Plewniak, F., Jeanmougin, F. and Higgins, D.G. (1997), The CLUSTAL_X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools, *Nucleic Acids Research*, 25, 4876-4882.

[19] Thompson, J.D., Higgins, D. G. and Gibson, T. J. (1994), CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, *Nucleic Acids Research*, 22, 4673-4680.