A Q-learning Approach for Aligning Protein Sequences

Ioan-Gabriel Mircea, Gabriela Czibula, Maria-Iuliana Bocicor
Faculty of Mathematics and Computer Science
Babeş-Bolyai University
1, M. Kogălniceanu Street, 400084, Cluj-Napoca, Romania

***Abstract*—Protein multiple sequence alignment is significant in the field of bioinformatics as it may reveal important information about the protein sequences' functional, structural or evolutionary relationships. It involves the alignment of three or more biological protein sequences and represents a real challenge both from a biological and a computational point of view. Q-learning is a reinforcement learning technique in which an artificial agent learns to find an optimal sequence of actions to achieve a goal by receiving rewards for its chosen actions. This paper investigates a Q-learning based model for the multiple sequence alignment problem applied on protein sequences. The experimental evaluation of the model is performed on two artificial data sets and on benchmark problem sets selected from the BAliBASE database. The obtained results show the effectiveness of using reinforcement learning for determining the optimal alignment of multiple protein sequences.**

***Keywords*: Bioinformatics, Protein Multiple Sequence Alignment, Machine Learning, Q-learning.**

I. Introduction

Sequence analysis and comparison represent a significant component in biology research. Protein multiple sequence alignment (PMSA) refers to the process of aligning protein sequences in order to identify similar regions, which might give important information about evolutionary, structural and functional relationships between the proteins.

Pairwise protein alignment means aligning two proteins and dynamic programming is widely used to obtain such alignments. The same technique can be applied for multiple protein sequences, but the required time grows exponentially with the number of sequences and sequence lengths. We have previously proposed one such algorithm, which is based on reinforcement learning (RL) [1].

In this paper we aim to propose a Q-learning based model for the protein multiple sequence alignment problem. Q-learning is a reinforcement learning (RL) [2] based approach in which an agent can learn to behave in a certain way by receiving rewards (or reinforcements) for its chosen actions. The learner is not told which actions to take, but it has to learn which actions produce the highest cumulative reward.

Since the PMSA problem is NP-complete [3], it is hard to obtain the optimal solution with a reasonable execution time. Therefore, there is a continuous interest in developing heuristic approaches able to determine near optimal solutions in polynomial time. As far as we know, our approach is novel, since the PMSA problem has not been addressed in the literature using Q-learning, so far.

978-1-4673-8200-7/15/$31.00

The rest of the paper is organized as follows. Section II presents the main aspects related to the PMSA problem, emphasizing the importance of the problem as well as existing methods for solving it. The Q-learning based approach for solving the PMSA problem is introduced in Section III. Section IV provides several experiments performed using protein sequences. An analysis of the proposed method, as well as comparisons with similar work from the literature are given in Section V. Section VI presents the conclusions of the paper and indicates directions for future improvements.

II. Background

This section presents the protein multiple sequence alignment problem and its relevance in bioinformatics. We also offer a short revision of other techniques from the literature that address the protein multiple sequence alignment problem.

A. Protein Multiple Sequence Alignment (PMSA)

1) Problem definition and relevance: Protein multiple sequence alignment plays an important role in the field of bioinformatics as it may reveal important information about the protein sequences' functional, structural or evolutionary relationships. It involves the alignment of three or more biological protein sequences and is a real challenge both from a biological and a computational point of view. The aim is to find common patterns, to construct an alignment of two or more proteins so as to maximize the similarities between them. Pairwise alignment only involves two protein sequences.

Proteins are large macromolecules consisting of one or more long chains of amino acids. In order to assess the similarity between proteins they need to be aligned properly such that corresponding amino acids reside on the same column. This is achieved by inserting gaps in certain positions inside the sequences. Multiple protein sequences can be aligned at the same time with the purpose of highlighting similarities between them, uncovering motifs that lead to the secondary protein structure. The task of multiple protein alignment is highly delicate since improper alignments can lead to the misinterpretation of protein structure that may in turn lead to folding anomalies, inexact phylogeny, and incorrect functional interpretations of the proteins.

PMSA is most often needed for sequences that are assumed to be connected by evolutionary relationships. From the result

of such an alignment biologists can identify mutations that happened in proteins, sequence motifs or secondary structures.

The cost measure which is usually used for evaluating the quality of an alignment is the sum of pairs SP score method [4]. This score is obtained by summing the substitution scores of all possible pairwise combinations of aminoacid characters over all columns of a multiple protein sequence alignment:

$$SP = \sum_{i=1}^{N} \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} sc(a_i^j, a_i^k) \quad (1)$$

where: n is the number of protein sequences in the alignment; N is the number of columns in the alignment; $a_i, i \in \{1, \ldots, N\}$ represents column i from the alignment; $a_i^j, j \in \{1, \ldots, n\}$ is the j^{th} aminoacid character from the i^{th} column; $sc(a_i^j, a_i^k)$ indicates the comparing score between the characters a_i^j and a_i^k.

a) Example: Next, we illustrate PMSA by using a simple example, taken from [5]. Let us consider four protein sequences ($n = 4$):

S_1 = MQPILLL S_2 = MLRLL
S_3 = MKILLL S_4 = MPPVLIL

We consider a score of **-4** for the *gap penalty*, **-3** for the *mismatch penalty* and **+5** for the *match score*.

As it will be shown in Section IV-A1, an optimal alignment (having a maximum associated SP score) obtained from the considered sequences is given in Table I.

M	L	R	-	L	-	L
M	P	P	V	L	I	L
M	Q	P	I	L	L	L
M	K	-	I	L	L	L

TABLE I. EXAMPLE OF ALIGNMENT.

For this alignment we compute the sum of pairs score (denoted by sp_i for each column i, $1 \leq i \leq 7$). The value sp_i is computed as indicated in Formula (2).

$$sp_i = \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} sc(a_i^j, a_i^k) \quad (2)$$

The sum of pairs score sp_4 for the fourth column is -13 and is computed as

$$\begin{aligned} sp_4 &= \sum_{j=1}^{3} \sum_{k=j+1}^{4} sc(a_i^j, a_i^k) = \\ &= score(-, V) + score(-, I) + \\ &\quad score(-, I) + score(V, I) + \\ &\quad score(V, I) + score(I, I). \end{aligned} \quad (3)$$

The sp values for the columns, computed as indicated below, are shown in Table II.

Thus, the sum of pairs score for the alignment given in Table I is 33.

sp_1	sp_2	sp_3	sp_4	sp_5	sp_6	sp_7
30	-18	-13	-13	30	-13	30

TABLE II. COLUMN SCORES FOR THE ALIGNMENT INDICATED IN TABLE I .

2) Sequence to profile alignment: The well-known solution to the PMSA problem is based on the dynamic programming method for aligning the amino acid sequences of two proteins (such as the Needleman-Wunsch [6] algorithm). This method is further extended to align multiple protein sequences. The main idea of the Needleman-Wunsch algorithm for aligning two proteins is to construct the best alignment by using optimal alignments of smaller subsequences, using the dynamic programming method. The method divides the problem of finding the optimal alignment of the full protein sequences into a series of smaller problems (i.e. optimal alignments of protein subsequences) and uses the solutions to the smaller problems to construct the solution to the initial problem.

In the case of aligning two proteins the optimal alignment score can be easily computed using the dynamic programming approach, considering the sum of pairs score method [4]. When aligning multiple amino acid sequences of proteins, a usual method is to progressively align a protein with a list of previously aligned proteins. This method is referred in the literature as the *sequence to profile* alignment [7].

In order to align a protein to a list of already aligned proteins, the Needleman-Wunsch algorithm is adapted to calculate the score of each column in the protein sequence against all the corresponding amino acid symbols in the list of already aligned proteins. In this approach, the profile of the partial alignment is computed [7]. The profile of a partial alignment keeps the frequencies of each amino acid in the alphabet for each position in the list of previously aligned proteins. Therefore, the comparison between two amino acid symbols in the Needleman-Wunsch algorithm can be replaced by a comparison between an amino acid symbol on a certain position in the protein and the column corresponding to the same position in the profile (representing the partial alignment).

This way, the optimal alignment *score* of the protein sequences can be progressively computed.

B. Related Work

In this section we present an overview of the existing methods for *protein multiple sequence alignment*. The reference approaches for the general *multiple sequence alignment* problem which are also used in the PMSA literature, are mentioned, as well.

As for the general MSA, a direct method for obtaining the globally optimal alignment of a sequence of proteins uses the dynamic programming technique [8]. The dynamic programming method is computationally very expensive, since the running times are growing exponentially with the size of the problem. In order to overcome this problem, various heuristics have been proposed in the PMSA literature for solving this problem.

Genetic algorithms (GA) have widely been used for solving the MSA problem (either for proteins or for DNA sequences),

with different variations, local search mechanisms or in conjunction with other machine learning techniques [9].

One of the most popular genetic algorithm approaches towards solving the Protein Multiple Sequence Alignment is the SAGA (Sequence Alignment by Genetic Algorithm) [10] which employs customized genetic operators for solving the MSA problem which include scheduling operators, ClustalW based operators for generating the initial population and specific crossover and mutation operators [11].

Chen et al. introduce in [12] a divide-and-conquer algorithm based on ant colony optimization which breaks the overall protein alignment problem into subproblems that concern the alignment of subsequences obtained by slicing the initial sequences vertically such that each alignment task is simplified. An ant colony approach is used to find the best alignment starting from each of the sequences and maximizing the alignment fitness function.

The ClustalW algorithm [13] is a progressive alignment method which performs a series of pairwise alignments on the set of sequences based on constructing a phylogenetic tree and favoring compact solutions by use of a dynamic gap penalty system.

Each progressive alignment method is strongly connected to the weighting scheme used to define the order in which pairwise alignments are performed. [14] proposes three such weighting schemes: one favoring local alignment, another global alignment and the latter reducing the bias of a global alignment taking into account local motifs by use of a double dynamic programming technique.

Since the PMSA problem is a particular case of MSA dealing with proteins, the secondary and tertiary structure of the proteins can be used in order to increase the accuracy of the alignments. More accurate scoring models can be obtained by studying the secondary structure of proteins as it was proposed in [15].Recent approaches from the PMSA literature focus on enhancing the performance of classic alignment techniques with additional protein related information, not only regarding secondary and tertiary structure. External protein metadata was successfully used in [16] on several protein benchmarks.

III. Our Q-learning based Approach for PMSA

Q-learning is an approach to reinforcement learning in which an artificial agent [17] learns to perform an optimal sequence of actions in order to achieve its goal, by learning the utilities of (state, action) pairs [18]. Thus, Q- learning represents a good approach for solving optimization problems.

We introduce in the following our Q-learning based approach for identifying an optimal alignment of a sequence of proteins.

A. The reinforcement learning task

Let us consider, in the following, that $\mathcal{P}$ is the input data set, consisting of n ($n > 1$) multi-dimensional protein sequences: $\mathcal{P} = (P_1, P_2, \ldots, P_n)$, each sequence P_i being a sequence of aminoacids, $P_i = (P_i^1, P_i^2, \ldots, P_i^{lg_i})$, P_i^j is an aminoacid character and lg_i represents the length of the protein P_i, i.e. the number of aminoacids in the protein sequence.

The *PMSA* problem consists of determining the alignment of the proteins from $\mathcal{P}$ which has the maximum associated score. From a computational point of view, the $PMSA$ problem can be viewed as the problem of generating a permutation $\tau = (\tau_1, \tau_2, \ldots \tau_n)$ of $\{1, 2, \ldots, n\}$ that maximizes the score $Score(\mathcal{P}_\tau)$ of the alignment of sequences considered in the order τ: $\mathcal{P}_\tau = (P_{\tau_1}, P_{\tau_2}, \ldots, P_{\tau_n})$. The score $Score(\mathcal{P}_\tau)$ of the alignment $\mathcal{P}_\tau$ represents the score computed as indicated in Section II-A .

The reinforcement learning task associated to the $PMSA$ problem will be further defined.

The *environment* (*state space*) $\mathcal{E}$ of the $PMSA$ agent will consist of $\frac{n^{n+1}-1}{n-1}$ states, i.e. $\mathcal{E} = \{es_1, es_2, ..., es_{\frac{n^{n+1}-1}{n-1}}\}$. The *initial state* of the agent in the environment is es_1. A state $s \in \mathcal{E}$ reached by the agent at a given moment after it started from the initial state is a *final* state if all the actions selected in its history represent a permutation of $\{1, 2, \ldots, n\}$. The environment of the $PMSA$ agent may be visualized as a tree, as it is depicted in Figure 1.

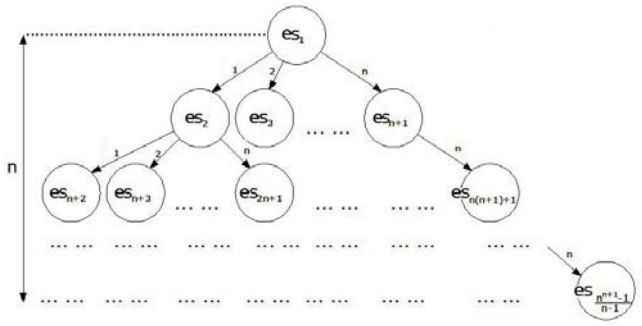

Fig. 1. The environment of the $PMSA$ agent.

The *action space* $\mathcal{A}ct$ of the $PMSA$ agent consists of n actions corresponding to the n possible values $1, 2, \ldots, n$ used to represent a permutation of $\{1, 2, \ldots, n\}$, i.e. $\mathcal{A}ct = \{act_1, act_2, \ldots, act_n\}$, where $act_i = i, \ \forall\, i \in \{1, \ldots, n\}$.

The *transition function* between the environment states $\phi : \mathcal{E} \to P(\mathcal{E})$ is defined as illustrated in Figure 1. More exactly, a given state $es \in \mathcal{E}$ has n *successor* (*neighbor*) states which are reached by the $PMSA$ agent by selecting the n possible actions from $\mathcal{A}ct$.

The *reinforcement function* will be defined below.

It is known that after a reinforcement learning process, the agent will learn to select those actions that maximize the discounted sum of rewards received on the path from the initial to a final state [17]. Thus, we will define the reinforcement function so as to obtain an alignment of the proteins having a maximum associated score.

Let us consider a path δ in the above defined environment strating from the initial state. If the sequence of actions selected on the path δ contains distinct actions, then we call the path to be *valid*. For an $n+1$-length *valid* path δ in the environment, the corresponding sequence of actions $a_1, a_2, \cdots, a_n$ can be viewed as a possible order for obtaining an alignment of the input protein sequences $\mathcal{P}$ (i.e. the alignment considered in the

order $P_{a_1}, P_{a_2}, \ldots, P_{a_n}$). The final alignment of the sequences of proteins is based on gradually aligning a protein to the profile corresponding to the previously aligned list of proteins (see Section II-A) and has an associated *score*, $\mathcal{S}core(\delta)$.

The $PMSA$ problem formulated as a reinforcement learning task will consist of training the agent to find a *valid* path δ from the initial to a final state having the maximum score $\mathcal{S}core(\delta)$ of the corresponding alignment.

We aim at obtaining a path δ having the maximum score computed by progressively aligning the sequences in the order given by the actions selected on the path. Thus, assuming that state δ_k is the current state of the agent and its history in the environment is $\delta = (\delta_0 = s_1, \delta_1, \delta_2, \ldots, \delta_{k-1})$ we define the reward obtained in state δ_k as follows.

- If k is 1, i.e the current state is the initial state, then the received reward is 0.
- If the path δ is not a *valid* one, then the agent receives a negative reward, -c, where c is a large positive constant.
- In all other situations, the reward received by the agent is the *optimal score* of the partial alignment given by the sequence of actions selected on the path δ. This score is computed using the algorithm for *sequence to profile* alignment (see Section II-A) and uses the match/mismatch score between two amino acid symbols x and y computed as in Formula (4).

$$sc(c_1, c_2) = \begin{cases} gap\,penalty & if\ c_1\ or\ c_2 =' \_' \\ match\,score & if\ c_1 = c_2 \\ mismatch\,score & otherwise \end{cases} \quad (4)$$

As defined below, the agent will receive a very small reward on paths that are not valid, thus it will learn to explore only valid paths. Considering the reward defined below, as the learning goal is to maximize the discounted sum of rewards received on a path from the initial to a final state, the agent is trained to find a valid path δ that maximizes the score of the corresponding alignment (i.e. $\mathcal{S}core(\delta)$). After the training of the $PMSA$ agent was completed, it is very likely that it will learn the optimal alignment of the sequences from $\mathcal{P}$.

B. The learning process

For training the *MSA agent*, we use the Q-learning algorithm [2], in which the agent learns an action value function, denoted by Q, function giving the expected utility of taking a given action in a given state [2]. The equation for Q-learning is given in Formula 5. In this equation, γ and α are, respectively, the discount factor for future rewards and the learning rate, $Q(s, a)$ denotes the value of doing the action a in state s, $r(s, a)$ denotes the reward received in state s after performing action a and s' represents the state of the environment reached by the agent after performing action a in state s

$$Q(s,a) = (1-\alpha)\cdot Q(s,a) + \alpha\cdot(r(s,a) + \gamma\cdot\max_{a'} Q(s',a')) \quad (5)$$

The idea of the training process is the following. After the Q values are initialized, during some training episodes, the agent will experiment (using an action selection mechanism) some (possible optimal) *valid* paths from the initial to a final state, updating the Q-values estimations according to the $Q-learning$ algorithm [18]. At the end of the training process, the Q-values estimations will be in the vicinity of the exact values.

The action selection mechanism used in the $Q-learning$ algorithm is the one introduced in [1] and uses a one step look-ahead procedure in order to guide the exploration of the search space: with probability $1-\epsilon$ choose the action act that maximizes the Q-value of the neighbor state; with probability ϵ^2, choose the action act that gives the maximum score obtained by aligning the protein sequence P_{act} with the profile corresponding to the alignment indicated by the current path; otherwise, select a random action. Since the alignment algorithm starts with two sequences, at the beginning of a training episode, we will use the following action selection mechanism: with probability $1-\epsilon$ select the actions that maximize the score of their alignment; otherwise select two random actions.

After the training of the agent has been finished, the optimal permutation is constructed by starting from the initial state and following the $Greedy$ mechanism until a solution is reached. From a given state, the agent transitions to a neighboring state having the maximum Q-value. It has been proven that the learned Q-values converge to their optimal values as long as all state-action pairs are visited an infinite number of times [19]. Consequently, the sequence of actions corresponding to the path δ learned by the $PMSA$ agent converges, in the limit, to the optimal alignment of the protein sequences, i.e. the alignment having the maximum associated score.

IV. Experimental Evaluation

The experimental evaluations of our Q-learning based approach for the protein MSA problem are presented in this section. Two artificial data sets and several benchmark problem sets selected from the BAliBASE database were used for our computational experiments. We have chosen BAliBASE data sets which have been previously used in the literature to allow us to compare our algorithm with related work. All the experiments presented in this section were carried out on a PC with an Intel Core i3 Processor at 2:4 GHz, with 3 GB of RAM. The information about the used data sets is synthesized in Table IV.

In all experiments we used the following parameter set for the Q-learning algorithm: the discount factor for the future rewards is $\gamma = 0.9$, so as to give more importance to future rewards; the learning rate is $\alpha = 0.8$ in order to obtain fast convergence and learning; the number of training episodes is $2 \cdot 10^4$; the look-ahead selection mechanism (Subsection III-B) was used with $\epsilon = 0.8$, to favor exploitation, but also allow exploration. Regarding the scoring matrix (gap penalty, mismatch penalty and mach score), the values we used vary from one experiment to another; they were chosen differently in each case to allow comparisons of our algorithm to other results reported so far.

The alignments obtained after applying our Q-learning algorithm are evaluated by three measures: the sum of pairs

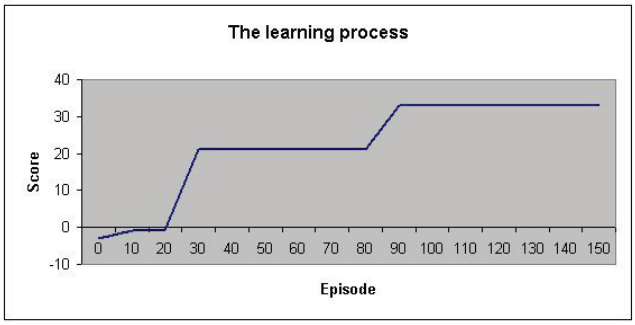

Fig. 2. The learning process for the first artificial data set

score [4], the number of exact matches (perfectly matched columns - EM) and the column score (CS), computed as the ratio between the number of columns that are exact matches and the total alignment length (AL). All these measures have to be maximized in order to obtain better alignments.

A. Artificial data sets

First we tested the capacity of our algorithm to find the optimal alignment on two small protein sequence data sets taken from [5]. For the scoring matrix the following values have been considered: **-4** for the *gap penalty*, **-3** for the *mismatch penalty* and **+5** for the *match score*. We have chosen these values mainly because they have been used in other works from the literature.

1) First artificial data set: The first artificial data set is composed of 4 protein sequences and was described in Section II-A1. Table III illustrates all possible permutations of the four protein sequences and the associated sum of pairs scores [4] obtained considered the given order. The order in which the proteins are considered in the alignment process is important, since the sequence to profile alignment algorithm obtains the profiles and then aligns the new sequences taking this order into account. We observe that there are several input orders that lead to the same maximum score.

For this example the state space consists of 341 states and the action space contains 4 actions, corresponding to the 4 sequences (Subsection III-A). Our Q-learning based algorithm obtains the optimal alignment $S_2S_4S_1S_3$ (illustrated in Table I), which has a sum of pairs score of 33 and 3 perfectly matched columns (meaning that all symbols on three columns are identical): columns 1, 5 and 7. The algorithm converges, on average, after less than 100 episodes (the average is computed over 5 runs) and the computational time needed for convergence is very low, less than 1 second.

Figure 2 depicts the Q-learning process for the first artificial data set that was described in this section. It can be seen how, during the training, the learned solution converges to the optimal one.

2) Second artificial data set: The second artificial data set used for evaluating our Q-learning proposal for PMSA is composed of 5 short sequences [5], which are in fact subsequences of real proteins:

S_1 = GAFTGEVSLPILKDFGVNWIVLGHSERRAYYGNEIVADKVAAAV
S_2 = GLASLKDFGVNWIVLGHSERRWYYGEVADKVAAAV
S_3 = GAFTGENSVDQIKDVGAKWVILGHSERRSEDDKFIADKTKFAL
S_4 = QAYTVSPVMLKDLGVTYVILGHSERRQMFAETDETVNKKVLAAF
S_5 = GSHTGHVLPEAVKEAGAVGTLLNHSENRMILADLEAAIRRAE

For this example the state space consists of 3906 states and the action space contains 5 actions, corresponding to the 5 sequences (Subsection III-A). The optimal alignment obtained by our algorithm is $S_1S_2S_3S_4S_5$, which has a sum of pairs score of -66 and 8 perfectly matched columns (meaning that all symbols on eight columns are identical): columns 13, 16, 23, 25, 26, 27, 29 and 47. The algorithm converges, on average, after less than 700 episodes (the average is computed over 5 runs) and the computational time needed for convergence is very low, less than 3 seconds. The alignment is illustrated in Figure 3.

```
GAFTGEVSLPILKDFGVNWIV-LGHSERR---AYYGNEIVADKVAAAV
-------GLASLKDFGVNWIV-LGHSERR---WYYG-E-VADKVAAAV
GAFTGENSVDQIKDVGAKW-VILGHSERR---SEDDKFI-ADKTKFAL
QAYT--VSPVMLKDLGVT-YVILGHSERRQMFAETD-ETVNKKVLAAF
GSHTGHVLPEAVKEAGAVGT-LLNHSENRMILA--DLEA-AIRR--AE
```

Fig. 3. The alignment obtained on the second artificial set.

B. BAliBASE benchmarks

Besides the synthetic data sets presented in the previous subsections, we tested our algorithm on ten benchmark problem sets selected from the BAliBASE 2.0 database [20]. BAliBASE [21] is a a well-known benchmark database consisting of reference multiple sequence alignments. The data sets which will be further considered in our evaluations were previously used in the PMSA literature in [12].

We ran our Q-learning algorithm for ten data sets from BAliBASE 2.0, namely: *1fmb*, *1tvxA*, *1ubi*, *2trx*, *2fxb*, *1krn*, *1uky*, *3grs*, *1havA* data sets and a subset of the *1aboA* data set. Details about these data sets are given in Table IV. We mention that from the *1aboA* data set the first 10 protein sequences were considered. For the experiments, the parameter setting described in Section IV is used. For the scoring matrix the following values have been considered: **-2** for the *gap penalty*, **-1** for the *mismatch penalty* and **+2** for the *match score*. We mention that the score between two gaps is considered **0**. These values were also considered in an existing work from the literature [12] and this allows us to compare the results of our algorithm with other results reported so far.

Data set	No. of sequences	Average length	Max. seq. length	Min. seq. length
First artificial data	4	6.25	7	5
Second artificial data	5	41.6	44	35
1fmb [20]	4	100	98	104
1tvxA [20]	4	60.5	69	51
1ubi [20]	4	81.75	94	76
2trx [20]	4	90.5	99	85
2fxb [20]	5	57.4	63	55
1krn [20]	5	78	82	66
1uky [20]	4	203	220	186
3grs [20]	4	224.75	237	201
1havA [20]	5	178	199	136
aboA [20]	10	59.1	66	57

TABLE IV. DETAILED INFORMATION OF THE DATA SETS USED FOR OUR EXPERIMENTS.

No.	Permutation	Score	No.	Permutation	Score	No.	Permutation	Score
1	$S_1S_2S_3S_4$	-3	**9**	$S_2S_3S_1S_4$	21	**17**	$S_3S_4S_1S_2$	-44
2	$S_1S_2S_4S_3$	-3	**10**	$S_2S_3S_4S_1$	13	**18**	$S_3S_4S_2S_1$	-27
3	$S_1S_3S_2S_4$	-3	**11**	$S_2S_4S_1S_3$	33	**19**	$S_4S_1S_2S_3$	-1
4	$S_1S_3S_4S_2$	-1	**12**	$S_2S_4S_3S_1$	33	**20**	$S_4S_1S_3S_2$	-1
5	$S_1S_4S_2S_3$	-1	**13**	$S_3S_1S_2S_4$	-3	**21**	$S_4S_2S_1S_3$	33
6	$S_1S_4S_3S_2$	-1	**14**	$S_3S_1S_4S_2$	-1	**22**	$S_4S_2S_3S_1$	33
7	$S_2S_1S_3S_4$	-3	**15**	$S_3S_2S_1S_4$	21	**23**	$S_4S_3S_1S_2$	-44
8	$S_2S_1S_4S_3$	-3	**16**	$S_3S_2S_4S_1$	13	**24**	$S_4S_3S_2S_1$	-27

TABLE III. ALL THE POSSIBLE PERMUTATIONS, AND THE ASSOCIATED SCORE FOR THE OBTAINED ALIGNMENT.

Data set	Score	AL	EM	CS	No. of epochs	Time (sec.)
First artificial data	33	7	3	0.429	< 100	< 1
Second artificial data	-66	48	8	0.167	< 700	< 3
1fmb [20]	246	108	28	0.259	< 100	< 1
1tvxA [20]	-300	76	4	0.053	150	< 1
1ubi [20]	-378	97	1	0.01	200	< 1
2trx [20]	-372	104	4	0.038	200	< 1
2fxb [20]	215	65	15	0.23	300	< 1
1krn [20]	102	85	21	0.247	300	1
1uky [20]	-900	237	11	0.046	200	4
3grs [20]	-1116	279	14	0.05	200	5
1havA [20]	-1738	222	3	0.014	2000	53
aboA [20]	-900	70	4	0.057	19400	371

TABLE V. RESULTS OBTAINED BY OUR Q-LEARNING ALGORITHM. IN THIS TABLE THE FOLLOWING ABBREVIATIONS ARE USED: AL - ALIGNMENT LENGTH, EM - EXACT MATCHES (NUMBER OF PERFECTLY MATCHED COLUMNS) AND CS - COLUMN SCORE ($CS = \frac{EM}{AL}$).

Table V shows the experimental results obtained by applying our Q-learning approach for PMSA on the two artificial data sets presented in Section IV-A as well as on the benchmark problem sets selected from the BAliBASE 2.0 database. For each data set we depict in Table V the score, the length of the obtained alignment, the number of matched columns, the column score, the number of epochs and the time needed by the Q-learning algorithm for convergence.

Considering the experimental results presented above, we notice that the computational time and number of epochs of the Q-learning algorithm are influenced by the size of the problem (which is proportional to the number and the average length of the sequences). It is worth mentioning that the sequences' length has impact mainly in the sequence to profile alignment algorithm and therefore to the total computational time of our algorithm, while the number of sequences is important in the reinforcement learning process, thus influencing both the number of epochs and the computational time.

V. DISCUSSION AND COMPARISON TO RELATED WORK

An analysis of the method proposed in this paper, as well as comparisons with similar approaches from the literature are provided in this section.

First, we compare the results of applying the Q-learning algorithm on the artificial data sets with the alignments given in [5].

From Table VI we observe that for the first artificial data set, our results are clearly better than the alignment from [5], considering all evaluation measures (Score, AL, EM, CS). For the second artificial data set, using the Q-learning algorithm we obtain an alignment which is very close (considering the score) to the alignment from [5]. Moreover, it has to be noted that the number of perfectly matched columns (8) is the same for both alignments.

In the following, we will discuss the results obtained by our Q-learning algorithm in comparison with the results reported in [12], for the same data sets from the BAliBASE database. We also compare the alignments obtained by our Q-learning method with the reference alignments provided by the BAliBASE database [21]. Table VII shows the results of these comparisons, considering the *score* of the alignments, the AL, EM and CS evaluation measures, as well as the running time needed to provide the solution. We have to mention that even if we have used the same scoring matrix as the one reported in [12], the scores reported in [12] for the reference alignments differ than the ones we have obtained. Thus, for the Divide-Ant-MSA approach we decided to present in Table VII the scores given in paper [12] scaled proportionally with respect to the score we have obtained for the reference alignment. We also note that in [12] only the score and the running time are reported as evaluation measures for the alignments.

From Table VII we notice that, for the first eight data sets, the solution quality of the Q-learning approach outperforms both the alignment provided by the Divide-Ant-MSA approach from [12] and the reference alignment [21] considering all evaluation measures: the score of the alignment, the number of exact matches, the column score and the running time. For the "1havA" data set, our result is better than the reference alignment considering the score, but is slightly worse considering the EM and CS measures. Considering the result from [12] for the "1havA" data set, our result is better considering the score of the obtained alignment.

Regarding the computational time, one can observe from Table VII that our Q-learning method obtains the solution in less time than the Divide-Ant-MSA algorithm from [12] in 8 out of 9 cases. It has to be noted that the exact configuration for the computer on which the experiments from [12] are performed is not given. The running time of Divide-Ant-MSA is shown in [12] to be about 1% to 5% of that of SAGA [10], a MSA application using an evolutionary algorithm. Thus, our Q-learning algorithm is much faster than SAGA . For example, for 4 sequences with length of 200 (like the "3grs" data set) the running time of SAGA is 200 to 400 seconds [12], while

Data set	Our RL approach				Alignment from [5]			
	Score	AL	EM	CS	Score	AL	EM	CS
First artificial data	33	7	3	0.429	25	7	2	0.286
Second artificial data	−66	48	8	0.167	−65	46	8	0.174

TABLE VI. COMPARATIVE RESULTS FOR THE DATA SETS CONSIDERED FOR EVALUATION. IN THIS TABLE THE FOLLOWING ABBREVIATIONS ARE USED: AL - ALIGNMENT LENGTH, EM - EXACT MATCHES (NUMBER OF PERFECTLY MATCHED COLUMNS) AND CS - COLUMN SCORE ($CS = \frac{EM}{AL}$).

Data set	Our approach					Reference alignment [21]				Divide-Ant-MSA [12]	
	Score	AL	EM	CS	Running time (s)	Score	AL	EM	CS	Score	Running time (s)
1fmb	246	108	28	0.259	<1	240	106	26	0.245	235	4
1tvxA	−300	76	4	0.053	< 2	−369	78	1	0.012	-344	2
1ubi	−378	97	1	0.01	< 1	−483	105	1	0.009	-411	3
2trx	−372	104	4	0.038	< 1	−450	108	1	0.009	-465	3
2fxb	215	65	15	0.23	< 1	212	65	15	0.23	203	4
1krn	102	85	21	0.247	1	132	82	21	0.256	120	5
1uky	−900	237	11	0.046	4	−1191	261	6	0.022	-1213	12
3grs	−1116	279	14	0.05	5	−1188	266	6	0.022	-1215	13
1havA	−1738	222	3	0.014	53	−2080	245	5	0.02	-2096	19

TABLE VII. COMPARATIVE RESULTS FOR THE DATA SETS CONSIDERED FOR EVALUATION. IN THIS TABLE THE FOLLOWING ABBREVIATIONS ARE USED: AL - ALIGNMENT LENGTH, EM - EXACT MATCHES (NUMBER OF PERFECTLY MATCHED COLUMNS) AND CS - COLUMN SCORE ($CS = \frac{EM}{AL}$).

the running time of Q-learning is 5 seconds.

All the alignments that we have obtained with our Q-learning algorithm for PMSA have also been compared to alignments obtained with 2 different tools often used by biologists, BioEdit [22] (currently used by the biologist team we are working with [23]) using ClustalW [24] (which aligns the sequences by constructing a phylogenetic tree) and MAFFT [25] (which employs fast Fourier transform). BioEdit and MAFFT are using for alignment the Blosum62 scoring matrix [26]. Thus, we also ran our Q-learning method using the same scoring matrix. Table VIII illustrates the comparative results for all the data sets that we experimented on. It can be observed that our approach is better than BioEdit in **7** cases, and performs worse in **5** cases, in terms of sum of pairs score. Considering the same evaluation measure, our algorithm outperforms MAFFT in **9** cases and has a score with a smaller value in **3** cases. Regarding the column score measure, we remark that the ratio of columns that match perfectly is better for our approach, except for 5 cases in which it is slightly worse (and this is generally caused by the length of the obtained alignment and not by the number of exact matches). Besides these five cases, the column score for our algorithm is greater in **9** cases than the one returned by MAFFT. Compared to the column score of the alignments retrieved by BioEdit, the columns score of the RL algorithm is better in **9** cases and equal in **1** case. As conclusion of these comparisons, considering the sum of pairs score and the column score measures, out of **48** comparisons, our approach is better or equal than BioEdit and MAFFT in **35** cases and slightly worse in **13** cases.

Based on the feedback received from the biologists [23], the CS measure was mentioned as highly relevant in the evaluation process of an MSA algorithm since it favors alignments of less length and greater column matching scores. The comparisons of our Q-learning approach for MSA against other approaches in the literature (see Tables VI, VII and VIII) with respect to the CS measure reveal the following: out of **35** comparisons, our approach is better in **25** cases, equal in **2** cases and slightly worse in **8** cases. These results are depicted in Figures 4 and 5 and indicate the effectiveness of our proposal.

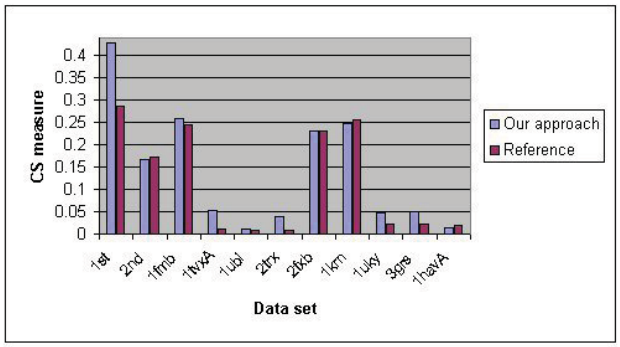

Fig. 4. Comparative results with the reference alignments.

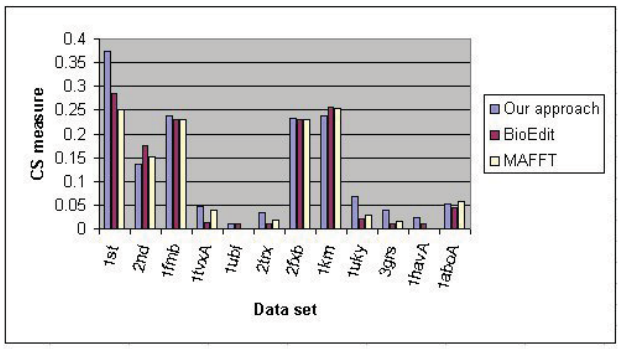

Fig. 5. Comparative results with BioEdit and MAFFT.

VI. CONCLUSIONS AND FUTURE WORK

We have introduced in this paper a Q-learning based approach for solving the protein multiple sequence alignment problem. The Q-learning algorithm was used for learning the optimal policy of the agent in its environment, which is very likely to correspond to the optimal alignment of the amino acid sequences of proteins.

Experiments were carried out on two synthetic data sets

Data set	Our RL approach				BioEdit [22]				MAFFT [25]			
	Score	AL	EM	CS	Score	AL	EM	CS	Score	AL	EM	CS
First artificial data	62	8	3	0.375	46	7	2	0.286	14	8	2	0.25
Second artificial data	491	51	7	0.137	533	46	8	0.174	469	46	7	0.152
1fmb	1335	114	27	0.237	1357	105	24	0.229	1364	105	24	0.229
1tvxA	−279	87	4	0.046	−312	70	1	0.014	−435	100	4	0.04
1ubi	−333	109	1	0.01	−159	99	1	0.01	−1142	147	0	0
2trx	−76	116	4	0.034	−3	102	1	0.01	−33	108	2	0.019
2fxb	1481	69	16	0.232	1460	65	15	0.231	1471	65	15	0.231
1krn	1602	88	21	0.239	1558	82	21	0.256	1575	83	21	0.253
1uky	−433	282	19	0.067	−339	239	5	0.02	−815	274	8	0.029
3grs	−759	306	12	0.04	−785	245	2	0.01	−1387	317	5	0.016
1havA	−1661	257	6	0.023	−1810	210	2	0.01	−3991	278	0	0
aboA	2932	79	4	0.051	2803	68	3	0.044	3380	70	4	0.057

TABLE VIII. COMPARATIVE RESULTS FOR THE DATA SETS CONSIDERED FOR EVALUATION. IN THIS TABLE THE FOLLOWING ABBREVIATIONS ARE USED: AL - ALIGNMENT LENGTH, EM - EXACT MATCHES (NUMBER OF PERFECTLY MATCHED COLUMNS) AND CS - COLUMN SCORE ($CS = \frac{EM}{AL}$). THE BLOSUM62 SCORING MATRIX [26] WAS USED FOR OBTAINING THE ALIGNMENTS.

as well as on benchmark problem sets selected from the BAliBASE database. The obtained results and their comparison to the similar related work from the literature emphasize the effectiveness of using reinforcement learning for solving the PMSA problem.

Further work will be made in order to extend the experimental evaluation of the proposed Q-learning method on larger protein data sets, to better assess its performance. The application of other reinforcement learning algorithms, such as SARSA learning [27] may be further considered. We also aim at investigating an extension of the proposed reinforcement learning method to a *fuzzy* approach.

REFERENCES

[1] M. Bocicor, I. Mircea, and G. Czibula, "A novel reinforcement learning based approach to multiple sequence alignment," *Information Sciences*, vol. under review, 2014.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[3] L. Wang and T. Jiang, "On the complexity of multiple sequence alignment," *Comput Biol*, vol. 4, pp. 337–348, 1994.

[4] D. Lipman, S. Altschul, and J. Kececioglu, "A tool for multiple sequence alignment," *Proc. Natl. Acad. Sci. USA*, vol. 86, pp. 4412–4415, 1989.

[5] T. Taylor, "Multiple sequence alignment," http://binf.gmu.edu/ttaylor/MNPSTUTORIAL/msa.txt.

[6] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443 – 453, 1970.

[7] M. Marti-Renom, M. Madhusudhan, and A. Sali, "Alignment of protein sequences by their profiles," *Protein Science: A Publication of the Protein Society*, vol. 13, pp. 1071–1087, 2004.

[8] F. V. Nelwamondo, D. Golding, and T. Marwala, "A dynamic programming approach to missing data estimation using neural networks," *Information Sciences*, vol. 237, pp. 49–58, Jul. 2013.

[9] K. Li, S. Kwong, R. Wang, K.-S. Tang, and K.-F. Man, "Learning paradigm based on jumping genes: A general framework for enhancing exploration in evolutionary multiobjective optimization," *Information Sciences*, vol. 226, pp. 1–22, Mar. 2013.

[10] C. Notredame and D. G. Higgins, "SAGA: Sequence alignment by genetic algorithm," *Nucleic Acids Research*, vol. 24, no. 8, pp. 1515–1524, 1996.

[11] R. Thomsen and W. Boomsma, "Multiple sequence alignment using SAGA: Investigating the effects of operator scheduling, population seeding, and crossover operators," in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3005, pp. 113–122.

[12] Y. Chen, Y. Pan, J. Chen, W. Liu, and L. Chen, "Partitioned optimization algorithms for multiple sequence alignment," in *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, vol. 2, April 2006, pp. 5 pp.–.

[13] J. Thompson, D. G. Higgins, and T. J. Gibson, "ClustalW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucl. Acids Res.*, vol. 22, no. 22, pp. 4673–4680, 1994.

[14] J. Heringa, "Local weighting schemes for protein multiple sequence alignment," *Computers and Chemistry*, vol. 26, no. 5, pp. 459 – 477, 2002.

[15] J. Kececioglu, E. Kim, and T. Wheeler, "Aligning protein sequences with predicted secondary structure," *J Comput Biol.*, vol. 17, no. 3, pp. 561–580, 2010.

[16] L. Ait, E. Corel, and B. Morgenstern, "Using protein-domain information for multiple sequence alignment," in *2012 IEEE 12th International Conference on Bioinformatics Bioengineering (BIBE)*, Nov 2012, pp. 163–168.

[17] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[18] P. Dayan and T. Sejnowski, "Td(Lambda) converges with probability 1," *Mach. Learn.*, vol. 14, pp. 295–301, 1994.

[19] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[20] A. Bahr, J. Thompson, J.-C. Thierry, and O. Pocha, "Balibase (benchmark alignment database): enhancements for repeats, transmembrane sequences and circular permutations," *Nucleic Acids Res*, vol. 29, no. 1, pp. 323–326, 2001.

[21] J. Thompson, F. Plewniak, and O. Poch, "Balibase: a benchmark alignment database for teh evaluation of multiple alignment programs," *Bioinfomratics Applications Note*, vol. 15, no. 1, pp. 87–88, 1999.

[22] "Biological sequence alignment editor," http://www.mbio.ncsu.edu/bioedit/bioedit.html.

[23] "Institute of Interdisciplinary Research in Bio-Nano-Sciences," http://bionanosci.institute.ubbcluj.ro/.

[24] M. Larkin, G. Blackshields, N. Brown, R. Chenna, P. McGettigan, M. H., F. Valentin, I. Wallace, A. Wilm, R. Lopez, J. Thompson, T. Gibson, and D. Higgins, "ClustalW and ClustalX version 2.0," *Bioinformatics*, vol. 23, no. 21, pp. 2947–2948, 2007.

[25] S. Katoh, "MAFFT multiple sequence alignment software version 7: improvements in performance and usability," *Molecular Biology and Evolution*, vol. 30, pp. 772–780, 2013.

[26] T. Taylor, "Blosum62 scoring matrix," http://www.ncbi.nlm.nih.gov/Class/FieldGuide/BLOSUM62.txt.

[27] A. Prez-Uribe and E. Sanchez, "A comparison of reinforcement learning with eligibility traces and integrated learning, planning and reacting," in *Concurrent Systems Engineering Series*. IOS Press, 1999.