# A Reinforcement Learning Based Approach to Multiple Sequence Alignment

Ioan-Gabriel Mircea(✉), Iuliana Bocicor(✉), and Gabriela Czibula(✉)

Faculty of Mathematics and Computer Science, Babeş-Bolyai University, 1, M. Kogălniceanu Street, 400084 Cluj-Napoca, Romania
{mircea,iuliana,gabis}@cs.ubbcluj.ro

**Abstract.** Multiple sequence alignment plays an important role in comparative genomic sequence analysis, being one of the most challenging problems in bioinformatics. This problem refers to the process of arranging the primary sequences of DNA, RNA or protein to identify regions of similarity that may be a consequence of functional, structural or evolutionary relationships between the sequences. In this paper we tackle multiple sequence alignment from a computational perspective and we introduce a novel approach, based on reinforcement learning, for addressing it. The experimental evaluation is performed on several DNA data sets, two of which contain human DNA sequences. The efficiency of our algorithm is shown by the obtained results, which prove that our technique outperforms other methods existing in the literature and which also indicate the potential of our proposal.

**Keywords:** Bioinformatics · Multiple Sequence Alignment · Machine Learning · Reinforcement Learning

## 1 Introduction

Sequence alignment [9] plays an important role in molecular sequence analysis, being a key element in understanding the evolution and functioning of living organisms. As the genomes of an increasing number of organisms are being sequenced nowadays, there is a constant need to study and investigate the evolutionary, structural and functional relationships between these genomes. The aim is to find common patterns between sequences of DNA, RNA or proteins so as to construct an alignment of two or more sequences. When two sequences are involved, it is called pairwise alignment, when three or more sequences are involved, it is called multiple sequence alignment. Since aligning each pair of sequences from a set of several does not reveal the common information of all the sequences, multiple sequence alignment is often necessary to find the sequences from the set that share an evolutionary lineage, are descendant from a common ancestor or that maintain certain domains or structure.

In this paper we approach, from a computational perspective, the problem of *multiple sequence alignment* (MSA). We introduce a novel approach for solving the MSA problem, based on reinforcement learning (RL) - an approach to machine intelligence in which an agent can learn an optimal sequence of actions which lead to a goal, by receiving rewards or reinforcements on its chosen actions [23]. We have previously conducted in [16] some preliminary experiments using the RL model for MSA, with the purpose of determining the best setting for our method. The best alignment algorithm and the best action selection policy which were decided in [16] will be used in this paper for introducing the RL based method for MSA. To our knowledge, the MSA problem for DNA sequences has not been addressed in the literature using reinforcement learning, so far.

The rest of the paper is organized as follows. Section 2 presents the main aspects related to the topics approached in this paper, the *MSA problem* and *reinforcement learning*. The RL model that we propose is introduced in Sect. 3. Section 4 provides several experiments, using several real life data sets containing DNA sequences. Comparisons between our approach and related work from the literature are given in Sect. 5. Section 6 outlines some conclusions of the paper and indicates future research directions.

## 2 Background

We begin by presenting the main aspects of the MSA problem and its relevance in bioinformatics. Then, we continue to briefly describe some of the existing techniques in the literature that approach this problem.

### 2.1 Problem Relevance

Multiple sequence alignment plays a very important role in comparative genomic sequence analysis, being one of the most challenging problems in bioinformatics. Sequence alignment refers to the process of arranging the primary sequences of DNA, RNA or protein to identify regions of similarity that may be a consequence of functional, structural or evolutionary relationships between the sequences. The aim is to find common patterns, to construct an alignment of two or more sequences so as to maximize the similarities between them. Pairwise alignment only involves two sequences. Still, in practice, situations in which three or more sequences need to be aligned often occur. In such cases, aligning only pairs of sequences is not sufficient, as this does not reveal common information for all the sequences in the data set. Therefore, MSA is necessary in order to find various significant information about the evolutionary, functional or structural relationships that might exist between several sequences.

MSA is most often needed for sequences that are assumed to be connected by evolutionary relationships, that may have descended from common ancestors. From the result of such an alignment, multiple pieces of information can be inferred: the sequences' evolutionary origins, mutations that happened in the DNA or in proteins, identification of conserved sequence motifs or secondary structures. In addition, MSA is one of the first necessary steps required in constructing phylogenetic trees or determining sequences' homologies.

A direct method for producing an MSA uses dynamic programming to identify the globally optimal alignment solution, one who has a maximum alignment score. One of the most popular cost measures to evaluate the quality of an alignment is the sum of pairs $SP$ score method [15]. This score sums the substitution scores of all possible pairwise combinations of sequence characters over all columns of a multiple sequence alignment:

$$SP = \sum_{i=1}^{N} \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} s(c_i^j, c_i^k) \tag{1}$$

where:

- $n$ is the number of sequences in the alignment;
- $N$ is the number of columns in the alignment;
- $c_i, i \in \{1, \ldots, N\}$ represents column $i$ from the alignment;
- $c_i^j, j \in \{1, \ldots, n\}$ is the $j^{th}$ character from the $i^{th}$ column;
- $s(c_i^j, c_i^k)$ is a measure indicating the comparing score between characters $c_i^j$ and $c_i^k$.

## 2.2 Literature Review

A direct method for producing an MSA uses the dynamic programming technique [19] to identify the globally optimal alignment solution. Still, this method is computationally very expensive, the running times are growing exponentially with the size of the problem (the MSA problem is NP-complete [25]). For this reason various heuristics have been proposed in the literature for solving this problem and most of the practical alignment algorithms are based on such heuristics that produce quasi-optimal alignments.

Chen and Lin [4] tackle the DNA MSA problem using genetic algorithms in conjunction with divide-and-conquer techniques to determine the best cut points for DNA sequences. They tested their approach using several DNA sequence sets and showed that it outperforms other methods existing in the literature. The same authors offered a different approach to the same problem, based on genetic simulated annealing techniques to choose the best cutting point set from a DNA sequences data set [5].

Nguyen *et al.* [20] introduce a parallel hybrid genetic algorithm for the protein MSA problem. They propose a new chromosome representation and corresponding new genetic operators and they use two kinds of local search heuristics. In addition to these, their method is extended to parallel processing.

Aside from the GA approaches, other solutions to the MSA problem have been proposed, which use methods from the field of machine learning. A partitioning approach combined with an ant colony optimization (ACO) method was used by Chen et al. [6] to solve the classical MSA problem. The experimental results of this study show that the algorithm can significantly reduce the running time and improve the solution quality on large-scale multiple sequence alignment benchmarks.

Rasmussen and Krink [22] propose to solve the MSA problem using hidden Markov models (HMMs), with a new particle swarm optimization (PSO) based training method. Nasser et al. [17] used fuzzy logic for approximate matching of sequences, because fuzzy logic allows tolerance of inexactness or errors in subsequence matching. They develop a prototype for a fuzzy assembler, that is designed to work with low quality data, an advantage that most existing techniques do not possess. The assembler was tested on sequences from two genome projects and the obtained results demonstrate that it can generate optimal assemblies.

A genetic algorithm based approach to solving the multiple sequence alignment problem was proposed by Agarwal in [1]. It uses custom genetic algorithm operators such as the Gap shuffle mutation in order to produce offsprings of improved fitness that may describe a better alignment with respect to a "sum-of-pair" based scoring frame using PAM and BLOSUM scoring matrices. Furthermore, the genetic algorithm approach has been used to enhance the performance of an ant colony model presented in [26], trained to find a better alignment by searching the solution inside a planar graph designed to model possible multiple sequence alignment configurations.

Adapting the crossover and mutation probabilities according to the specific characteristics of the population, the immune adaptive genetic algorithm proposed by Chao and Shuai in [3] manages to improve the performance of a classical genetic algorithm approach by enhancing individual diversity when population variety decreases and favoring the convergence towards a global optimum.

A great amount of research has also been conducted on finding which of the existing state of the art multiple sequence approaches performs best in practice [24] and also which protein or DNA benchmarks provide the most reliable sources of genetic material to be used in research. Caroll et al. propose in [2] a benchmark for DNA sequences generated on the tertiary structure of the proteins they encode. The proposed benchmark is successfully used in DNA alignment in order to test a genetic algorithm enhanced with self-organizing crossover and selection operations [21].

## 3 A *RL* Model for Multiple Sequence Alignment

In this section we introduce the reinforcement learning model proposal for solving the *MSA* problem. First, we discuss about *sequence to profile* alignment, since it will be further used in our RL based approach.

### 3.1 Sequence to Profile Alignment

Even thought the pairwise alignment of two sequences is a rather straightforward task that may be easily achieved by use of a dynamic programming algorithm, the situation tends to become complicated if three or more sequences need to be aligned globally with one another.

The main difficulty consists in aligning several pre-aligned sequences with a given sequence. When a sequence is to be aligned with a set of already aligned sequences, a symbol from the sequence is matched against a whole column of corresponding symbols from all the pre-aligned sequences.

In order to surmount this impasse, the alignment is no longer conducted between a sequence and a set of pre-aligned sequences but between the sequence to be aligned and a profile characterizing the set.

The profile is computed in the following manner: for each position in the set of pre-aligned sequences a list is maintained holding information about the frequency of the appearance of each symbol from the alphabet $\{A, C, G, T, \_\}$, where $A$, $C$, $G$, $T$ represent the 4 nucleotides and $\_$ represents a gap.

The algorithm we propose for finding the optimal alignment and its corresponding score extends the *Needleman-Wunsch* algorithm [18] for detecting the optimal alignment of two sequences.

## 3.2 The Proposed *RL* Approach for the *MSA* Problem

Let us consider, in the following, that $\mathcal{S}eq$ is the input data set, consisting of $n$ $(n > 1)$ multi-dimensional DNA sequences: $\mathcal{S}eq = (Seq_1, Seq_2, \ldots, Seq_n)$, each sequence $Seq_i$ being a sequence of nucleotides, $Seq_i = (Seq_i^1, Seq_i^2, \ldots, Seq_i^{l_i})$, $Seq_i^j \in \{A, T, G, C\}$ is a nucleotide and $l_i$ represents the length of sequence $Seq_i$, i.e. the number of nucleotides in the sequence.

Let us denote by $Align^{Seq}$ the matrix representing the optimal alignment of sequences from the list $Seq$ (considered in the order $Seq_1, Seq_2, \ldots, Seq_n$). Such an alignment means that the sequences are progressively aligned as follows: $Seq_1$ is aligned with $Seq_2$, then the resulted optimal alignment is aligned with $Seq_3$. The next step is to align the optimal alignment obtained so far with $Seq_4$ and the iterative process is continued until all the sequences $Seq_1, Seq_2, \ldots, Seq_n$ are optimally aligned.

As indicated above, the *MSA* problem consists of determining the alignment of the sequences from $Seq$ which has the maximum associated score. From a computational point of view, the $MSA$ problem can be viewed as the problem of generating a permutation $\sigma = (\sigma_1, \sigma_2, \ldots \sigma_n)$ of $\{1, 2, \ldots, n\}$ that maximizes the score $Score(Seq_\sigma)$ of the alignment of sequences considered in the order $\sigma$: $Seq_\sigma = (Seq_{\sigma_1}, Seq_{\sigma_2}, \ldots, Seq_{\sigma_n})$. The score of the alignment $Seq_\sigma$ represents the score of the corresponding matrix $Align^{Seq_\sigma}$ and it has to be maximized.

We define the RL task associated to the $MSA$ problem as follows.

The **state space** $\mathcal{S}$ (the $MSA$ agent's environment) will consist of $\frac{n^{n+1}-1}{n-1}$ states, i.e. $\mathcal{S} = \{s_1, s_2, ..., s_{\frac{n^{n+1}-1}{n-1}}\}$. The *initial state* of the agent in the environment is $s_1$. A state $s_{i_k} \in \mathcal{S}$ $(i_k \in \{1, \ldots, \frac{n^{n+1}-1}{n-1}\})$ reached by the agent at a given moment after it has visited states $s_{i_1}, s_{i_2}, s_{i_3}, ...s_{i_{k-1}}$ (here $s_{i_1}$ is the initial state $s_1$) and has selected actions $a_{i_1}, a_{i_2}, ...a_{i_{k-1}}$ is a *terminal* (final or goal) state if the number of states visited by the agent in the current sequence is $n + 1$ (i.e. $k = n + 1$) and all the selected actions $(a_{i_1}, a_{i_2}, ...a_{i_n}$ represent a permutation of $\{1, 2, \ldots, n\}$ (the selected actions are distinct), i.e. $a_{i_j} \neq a_{i_k}$, $\forall\, j, k \in \{1, \ldots, n\}\ j \neq k$.

The **action space** $\mathcal{A}$ consists of $n$ actions available to the problem solving agent and corresponding to the $n$ possible values $1, 2, \ldots, n$ used to represent a solution (permutation of $\{1, 2, \ldots, n\}$), i.e. $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, where $a_i = i$, $\forall\, i \in \{1, \ldots, n\}$. From a state $s \in \mathcal{S}$ the agent can move in $n$ successor states, by executing one of the $n$ possible actions.

A graphical representation of the states space and the transitions between the states are given in Fig. 1. The circles represent states and the transitions between states are indicated by arrows labeled with the action that leads the agent from a state to another.

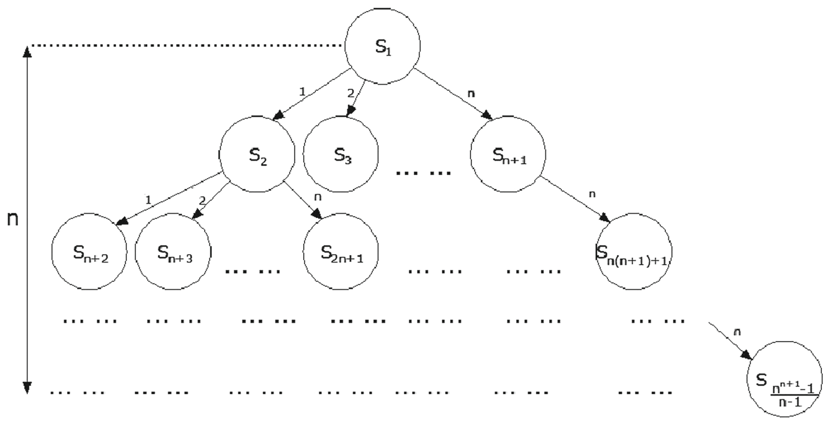

**Fig. 1.** Environment for the MSA agent

The **reward function** will be defined in the following (Formula (2)).

Let us consider a path $\pi$ in the above defined environment from the initial to a final state, $\pi = (\pi_0 \pi_1 \pi_2 \cdots \pi_n)$, where $\pi_0 = s_1$ and $\forall\, k \in \{0, \ldots, n-1\}$ the state $\pi_{k+1}$ is a *neighbor* of state $\pi_k$, i.e. $\pi_{k+1} \in \delta(\pi_k)$.

The sequence of actions obtained following the transitions between the successive states from the path $\pi$ will be denoted by $a_\pi = (a_{\pi_0} a_{\pi_1} a_{\pi_2} \cdots a_{\pi_{n-1}})$, where $\pi_{k+1} = \Delta(\pi_k, a_{\pi_k}),\ \forall\, k \in \{0, \ldots, n-1\}$. The sequence $a_\pi$ will be referred to as the *list of actions* associated to the path $\pi$.

The *list of actions* associated to a path $\pi$ gives us a list of sequences $\mathcal{S}eq_\pi = (Seq_{a_{\pi_0}}, Seq_{a_{\pi_1}}, \ldots, Seq_{a_{\pi_{n-1}}})$. If all the actions from the *list of actions* $a_\pi$ associated to a path $\pi$ are distinct (i.e. $a_{\pi_i} \neq a_{\pi_j}\ \forall 0 \leq i < j \leq n-1$), then $a_\pi$ is a permutation of $\{1, 2 \ldots n\}$ and can be viewed as a possible order for obtaining an alignment of the input sequences $Seq$ (i.e. the alignment considered in the order $Seq_{a_{\pi_0}}, Seq_{a_{\pi_1}}, \ldots, Seq_{a_{\pi_{n-1}}}$). Consequently, the value $\mathcal{S}core(\mathcal{S}eq_{a_\pi})$ corresponding to a path $\pi$ whose associated *list of actions* contains distinct actions would represent the optimal score of aligning the list of input sequences in the order given by $a_\pi$, i.e. in the order $Seq_{a_{\pi_0}}, Seq_{a_{\pi_1}}, \ldots, Seq_{a_{\pi_{n-1}}}$. Let us denote by $M(k,r)=(m_{i,j})_{1\le i\le k, 1\le j\le r}$ $(1 < k \le n)$ the matrix representing the optimal alignment of the list of sequences $(Seq_{a_{\pi_0}}, Seq_{a_{\pi_1}}, \ldots, Seq_{a_{\pi_{k-1}}})$ ($k$ denotes the number of rows and $r$ the number of columns of the alignment).

The $MSA$ problem formulated as a $RL$ task will consist of training the agent to find a path $\pi$ from the initial to a final state having the maximum score $\mathcal{S}core(\mathcal{S}eq_{a_\pi})$ of the corresponding alignment. After the reinforcement learning process, the agent will learn to execute those transitions (actions) that maximize the sum of rewards received on the path from the initial to a final state.

Since we aim at obtaining a path $\pi$ having the maximum score computed by progressively aligning the sequences in the order corresponding to the associated *list of actions* $a_\pi$, we define the reinforcement function as indicated in Formula (2).

$$r(\pi_k|\pi_{k-1},...\pi_0) = \begin{cases} 0 & if \quad k=1 \\ -\infty & a_{\pi_{k-1}} \in \{a_{\pi_0}, a_{\pi_1}, \ldots, a_{\pi_{k-2}}\} \\ \sum_{j=1}^{r}\sum_{l=1}^{k-1} s(m_{k,j}, m_{l,j}) & otherwise \end{cases} \tag{2}$$

In Formula (2) $r(\pi_k|\pi_{k-1},...\pi_0)$ denotes the reward received by the agent in state $\pi_k$, after its history in the environment is $\pi = (\pi_0 = s_1, \pi_1, \pi_2, \ldots, \pi_{k-1})$ and $s(x, y)$ represents the match/mismatch score between two characters $x$ and $y$.

The agent receives a negative reward on paths that are not valid (the associated *list of actions* contains actions that are not distinct), therefore it will learn to explore only valid paths. Considering the reward defined in Formula (2), as the learning goal is to maximize the total amount of rewards received on a path from the initial to a final state, the agent is trained to find a valid path $\pi$ that maximizes the score of the **alignment** that is indicated by its corresponding *list of actions* $a_\pi$.

Considering the way the reward was defined, it is very likely that after the training of the $MSA$ agent was completed, it will learn the optimal alignment of the sequences $Seq_1, Seq_2, \ldots, Seq_n$. Once the alignment is obtained, the optimal score of the alignment is computed.

During the training step of the learning process, the agent will determine its *optimal policy* in the environment, i.e. the mapping from states to actions that maximizes the sum of the received rewards.

For training the *MSA agent*, we propose a $Q$-learning approach, 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 [23]. 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 [8]. At the end of the training process, the $Q$-values estimations will be in the vicinity of the exact values.

The action selection mechanism we have used is the $\epsilon$-Greedy mechanism extended with a look-ahead step in order to guide the exploration of the search space. The look-ahead step consists in selecting, with a certain probability, the action $a$ that maximizes the score of aligning the sequence $Seq_a$ with the profile corresponding to the alignment associated to the *list of actions* of the current path.

## 4 Computational Experiments

This section presents experimental evaluations of our RL based model for the MSA problem. For the computational experiments developed in order to test our approach, we used several data sets, consisting of real DNA sequences. These data sets have previously been used in the literature to test other approaches to the MSA problem, which gives us the opportunity to compare our algorithm with related work. The information about all the public data sets is synthesized in Table 1. 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.

**Table 1.** Detailed information of the data sets used for our experiments.

| Data set | No. of sequences | Average length | Max. seq. length | Min. seq. length |
|---|---|---|---|---|
| Hepatitis C virus | 10 | 211.9 | 212 | 211 |
| Papio anubis | 5 | 1093 | 1098 | 1088 |
| 469 from oxbench_mdsa_all [2] | 3 | 332 | 339 | 321 |
| 429 from oxbench_mdsa_all [2] | 12 | 171 | 183 | 153 |
| Data set 1 from [26] | 3 | 39.33 | 40 | 39 |
| Lemur, gorilla, mouse data set from [26] | 3 | 93 | 94 | 92 |
| Rat, lemur, opossum data set from [26] | 3 | 129 | 129 | 129 |
| Mitochondrial human DNA - first data set | 8 | 365.75 | 372 | 363 |
| Mitochondrial human DNA - second data set | 6 | 362.83 | 366 | 360 |

### 4.1 Parameters Setting and Implementation Details

In our experiments, for applying the RL based approach introduced in Sect. 3, we have used a software framework that we have previously introduced for solving combinatorial optimization problems using reinforcement learning techniques [7].

The MSA agent was trained using the following parameter setting: the discount factor for the future rewards is $\gamma = 0.9$; the learning rate is $\alpha = 0.8$;

the number of training episodes is $10^4$; the action selection mechanism was used with $\epsilon = 0.8$. We are considering the following values for the scoring matrix: $\mathbf{-2}$ for the *gap penalty*, $\mathbf{-1}$ for the *mismatch penalty* and $\mathbf{+2}$ for the *match score*. We have chosen these values mainly because they have been used in other works from the literature and this allows us to compare the results of our algorithm with other results reported so far. The resulted alignments are evaluated by three measures: the sum of pairs score [15], the number of perfectly matched columns, or exact match (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 evaluation measures have to be maximized in order to obtain better alignments.

## 4.2 Hepatitis C Virus DNA Sequences

The first real life DNA sequence data set that we used contains 10 sequences belonging to the organism *Hepatitis C virus* (subtype 1b). These sequences were taken from the EMBL database [12], they have lengths of either 211 or 212 nucleotides and they can be retrieved by the following European Nucleotide Archive (ENA) accession numbers: Z75841, Z75850, Z75854, Z75852, Z75858, Z75851, Z75853, Z75859, Z75860, Z75861.

We ran our RL algorithm for this data set, using the parameter setting described in Sect. 4.1. The algorithm converges towards the solution rapidly, in less than 2 s in this case. The obtained alignment has a score of 18627 and the order of sequences is the one in which they were specified in the previous paragraph. The number of matched columns is 198 and considering that the sequences' length is approximately 212, we may state that this number is quite high. Table 2 shows the score, number of matched columns, the number of epochs and the time needed for convergence. We mention that the last two metrics (number of epochs and time) are averaged over 5 runs of the algorithm.

**Table 2.** Results for the Hepatitis C virus (subtype 1b), the Papio anubis data sets [12], for data sets 469 and 429 from oxbench_mdsa_all [2] and for three different species data sets [26] and for the two mitochondrial human DNA data sets, obtained by our RL based algorithm. In this table the following abbreviations are used: AL - alignment length, EM - exact matches (number of perfectly matched columns) and CS - column score ($CS = \frac{EM}{AL}$).

| Data set | Score | AL | EM | CS | No. of epochs | Time (sec.) |
|---|---|---|---|---|---|---|
| Hepatitis C virus | 18627 | 212 | 198 | 0.93 | <100 | <2 |
| Papio anubis | 18719 | 1110 | 918 | 0.82 | 160 | 190 |
| 469 from oxbench_mdsa_all | 565 | 378 | 171 | 0.45 | <100 | <1 |
| 429 from oxbench_mdsa_all | 8668 | 205 | 43 | 0.21 | 2950 | 186 |
| Data set 1 from [26] | 167 | 42 | 31 | 0.73 | <100 | <1 |
| Lemur, gorilla, mouse data set from [26] | 345 | 99 | 66 | 0.66 | <100 | <1 |
| Rat, lemur, opossum data set from [26] | 486 | 133 | 87 | 0.65 | <200 | <1 |

### 4.3 Papio Anubis (Olive Baboon) DNA Sequences

We tested our algorithm on a second publicly available data set, containing 5 DNA sequences belonging to the organism *Papio anubis* (olive baboon). Similar to the prior data set, the sequences were taken from the EMBL database [12], using the following ENA accession numbers: U35624, U35625, U35626, U35627, U35628. The average length of these sequences is 1093. We observe that in this case the number of sequences is smaller, but their lengths are approximately 5 times greater than in the previous case.

We used the parameter setting described in Sect. 4.1 to run our algorithm. Our RL algorithm obtained an alignment with a score of 18719 and the permutation of sequences is $1, 3, 2, 5, 4$, if the sequences are considered to be numbered in the order they were specified above. As in the previous case, the number of matched columns, 918 is high, considering the average length of the sequences (1093). The number of epochs and the computational time until convergence to the optimum solution are averaged over 5 runs of the algorithm and they are shown in Table 2, together with the sum of pairs score and the number of matched columns.

### 4.4 Data Set 469 from oxbench_mdsa_all

For the third experiment, we used the data set 469 from oxbench_mdsa_all [2], which contains three DNA sequences. The average length of the sequences is 332. Using the same parameter setting that we described in Sect. 4.1, our algorithm obtained an alignment with a score of 565 and the permutation of sequences $0, 2, 1$, considering the sequences in the order they are presented in the file *469.afa* from the above mentioned data set. The resulted alignment contains 378 columns and the number of perfectly matched columns is 171. Just as for the other data sets, the results are also shown in Table 2, averaged over 5 runs of the algorithm.

### 4.5 Data Set 429 from oxbench_mdsa_all

For the fourth experiment, we used another data set from the oxbench_mdsa_all [2] database: data set 429, which contains twelve DNA sequences. The average length of the sequences is 171. Using the same parameter setting that we described in Sect. 4.1, our algorithm obtained an alignment with a score of 8668. The resulted alignment contains 205 columns and the number of perfectly matched columns is 43. The results are shown in Table 2.

### 4.6 DNA Belonging to Three Different Species

The fifth experiment focuses on three data sets that were used for evaluating an ant colony with genetic algorithm based on planar graph [26]. Each one contains three sequences: the first data set contains shorter sequences, while the last two have sequence segments belonging to three kind of species. To be able to compare the results obtained by our RL based algorithm to the ones obtained by the ant colony genetic algorithm [26], we used as values for the scoring matrix the same values used in the previously mentioned work. These values are identical to the ones presented in Sect. 4.1, with only one observation: the alignment score of two gaps is 0.

**First Data Set.** The first data set was used in *Experiment 1* of [26]. For the exact DNA sequences, we invite the reader to consult the paper [26]. The average length of the sequences is 39. The score of the alignment obtained by our algorithm is 167, the alignment length is 42 and the number of matched columns is 31. The optimum permutation is $0, 1, 2$, considering the sequences in the order they are given in [26].

**Second Data Set: Lemur, Gorilla, Mouse DNA.** The second data set was used in *Experiment 2* of [26] and it contains DNA sequences from three species: lemur, gorilla and mouse. For the exact DNA sequences, we invite the reader to consult the paper [26]. The average length of the sequences is 93. The score of the alignment obtained by our algorithm is 345, the alignment length is 99 and the number of matched columns is 66. The optimum permutation is $0, 1, 2$, considering the sequences in the order they are given in [26].

**Third Data Set: Rat, Lemur, Opossum DNA.** The third data set was used in *Experiment 3* of [26] and it contains the third exon sequences of the beta-globin gene from three species: rat, lemur and opossum. For the exact DNA sequences, we invite the reader to consult the paper [26]. The average length of the sequences is 129. The score of the alignment obtained by our algorithm is 486, the alignment length is 133 and the number of matched columns is 87. The optimum permutation is $0, 1, 2$, considering the sequences in the order they are given in [26].

## 5 Comparison to Related Work

Most works in the literature that approach the MSA problem use protein data sets for evaluation. This is most probably due to the fact that there are considerably more multiple protein sequence alignment benchmarks (i.e. BAliBASE, OXBench, PREFAB and SMART) than DNA benchmarks, making the evaluation of MSA programs difficult for DNA sequences [2].

For our comparisons, we selected four papers presenting MSA approaches for the DNA sequence alignment [4,5,21,26]. The experimental evaluation of these methods are made on six of the data sets we used in Sect. 4: DNA sequences from the organisms *hepatitis C virus* and *papio anubis*, the data set 469 from *oxbench_mdsa_all* [2] and three other data sets with DNA belonging to different species.

In the following, we will discuss the results obtained by our RL based algorithm in comparison with other results reported in the literature, for the same data sets.

**Table 3.** Comparative results for the Hepatitis C virus (subtype 1b) and the Papio anubis data sets [12].

| Data set | Our RL approach | | GA + Divide and Conquer [4] | | Genetic Simulated Annealing [5] | |
|---|---|---|---|---|---|---|
| | Score | Exact match | Score | Exact match | Score | Exact match |
| Hepatitis C virus | 18627 | 198 | 17768 | 193 | 18287 | 194 |
| Papio anubis | 18719 | 918 | 17912 | 908 | 18030 | 914 |

**Table 4.** Comparative results for the data sets containing DNA belonging to three different species. 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 RL approach | | | | Ant Colony with GA [26] | | | |
|---|---|---|---|---|---|---|---|---|
| | Score | AL | EM | CS | Score | AL | EM | CS |
| Data set 1 from [26] | 167 | 42 | 31 | **0.73** | 167 | 42 | 31 | 0.73 |
| Lemur, gorilla, mouse data set from [26] | 345 | 99 | 66 | **0.66** | 348 | 96 | 62 | 0.64 |
| Rat, lemur, opossum data set from [26] | 486 | 133 | 87 | **0.65** | 471 | 131 | 84 | 0.64 |

First the alignments obtained by our method were compared to the alignments obtained by two other methods in the literature: [4,5], for the two data sets containing DNA sequences from the organisms *hepatitis C virus* and *papio anubis*. We note that our RL based algorithm proved to outperform the other approaches for the considered data sets. Table 3 shows the results obtained by our RL algorithm, in comparison with other results obtained by Chen *et al.* [4,5]. We notice that the RL approach proves to be more efficient, as the sum of pairs score of the alignment, as well as the number of matched columns have greater values that the ones reported by the approaches based on genetic algorithms (GA): the alignment method based on genetic algorithms and divide-and-conquer techniques [4] and the method based on genetic simulated annealing techniques [5]. We remark that the results we use for comparison from [5] are the best results reported in this paper, for the specified data set.

Another approach for the MSA problem, using an ant colony with genetic algorithm based on planar graph is presented in [26]. The authors experiment

**Table 5.** Comparative results for the data set 469 from oxbench_mdsa_all [2]. 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 RL approach | | | Self-Organizing GA [21] | | |
|---|---|---|---|---|---|---|
| | AL | EM | CS | AL | EM | CS |
| Data set 469 from oxbench_mdsa_all [2] | 378 | 171 | **0.45** | 406 | 45 | 0.11 |

**Table 6.** Comparative results for all the data sets: our RL algorithm is compared to ClustalW [14] and to MAFFT [13]. 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 RL approach | | | | ClustalW [14] | | | | MAFFT [13] | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Score | AL | EM | CS | Score | AL | EM | CS | Score | AL | EM | CS |
| Hepatitis C virus | 18627 | 212 | 198 | **0.93** | 18627 | 212 | 198 | 0.93 | 18627 | 212 | 198 | 0.93 |
| Papio anubis | 18719 | 1110 | 918 | **0.82** | 18827 | 1098 | 905 | 0.82 | 18860 | 1098 | 907 | 0.82 |
| 469 from oxbench_mdsa_all | 565 | 378 | 171 | **0.45** | 464 | 359 | 134 | 0.37 | 549 | 342 | 122 | 0.35 |
| 429 from oxbench_mdsa_all | 8668 | 205 | 43 | **0.20** | 9575 | 186 | 35 | 0.18 | 10218 | 183 | 35 | 0.19 |
| Data set 1 from [26] | 167 | 42 | 31 | **0.73** | 149 | 42 | 28 | 0.66 | 134 | 41 | 25 | 0.60 |
| Lemur, gorilla, mouse data set [26] | 345 | 99 | 66 | **0.66** | 345 | 94 | 59 | 0.62 | 345 | 94 | 59 | 0.62 |
| Rat, lemur, opossum data set [26] | 486 | 133 | 87 | **0.65** | 480 | 130 | 83 | 0.63 | 471 | 129 | 81 | 0.62 |

on three DNA data sets and we used these data sets in our experiments as well, to allow the comparison of the results (Sect. 4.6). Table 4 illustrates the results of our RL based algorithm in comparison with the algorithm described in the previously mentioned work. We observe that for the first data set the score obtained by our algorithm, as well as the alignment length and the number of columns that match perfectly are the same as those previously obtained in the literature. For the second data set, however, we notice that even if the score is lower and the length of the alignment is greater, the number of perfectly matched columns, as well as the column score are slightly better. As for the third data set, the results obtained by our algorithm outperform the ones reported in [26]: all considered evaluation measures (score, exact match and column score) indicate that our RL based approach leads to a better alignment.

We also compared our algorithm to the approach based on a self organizing genetic algorithm presented in [21]. The authors of this paper use the DNA data set 469 from oxbench_mdsa_all to evaluate their technique, therefore we also made experiments on this data set (Sect. 4.4). The comparative results are presented in Table 5. In this table we only show the alignment length, exact match and column score, as only the values of these evaluations measures are presented in the paper [21]. We remark that our algorithm clearly outperforms the one presented in [21], as the alignment length is shorter (meaning less gaps introduced), the number of columns that match perfectly is almost four times greater, these two measures inferring that the column score is also much better.

All the alignments that we have obtained with our algorithm have also been compared to alignments obtained with 2 different tools often used by biologists, both tools available via web-services on the EMBL-EBI website [10]. The tools in question are ClustalW [14] (which aligns the sequences by constructing a phylogenetic tree) and MAFFT [13] (which employs fast Fourier transform). Table 6 illustrates the comparative results for all the data sets that we experimented on. The comparative results considering the CS and EM measures are depicted in Figs. 2 and 3.

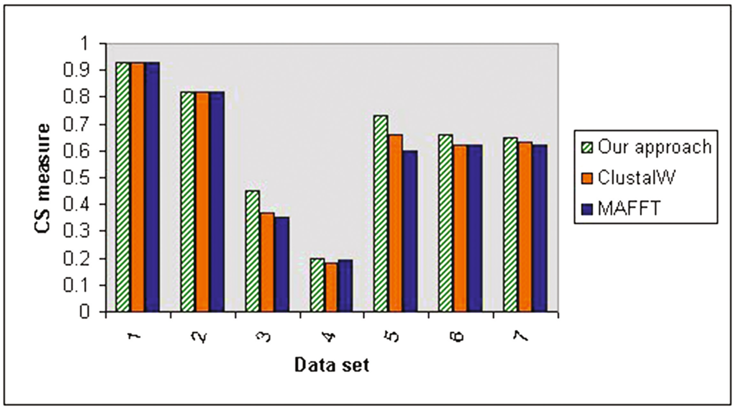

**Fig. 2.** Comparative results for all the data sets: our RL algorithm is compared to ClustalW [14] and to MAFFT [13] using the CS (column score) measure

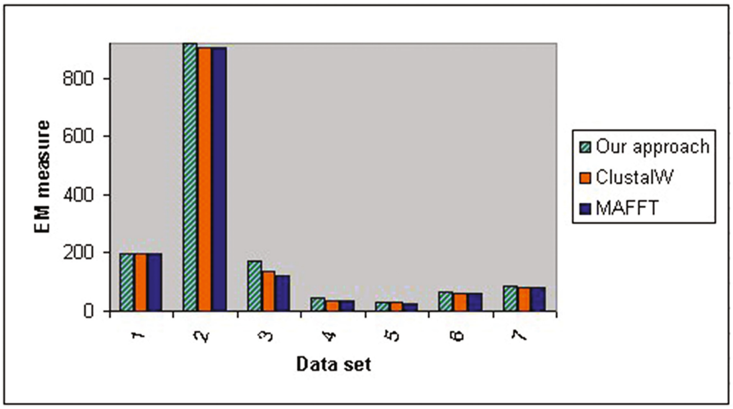

**Fig. 3.** Comparative results for all the data sets: our RL algorithm is compared to ClustalW [14] and to MAFFT [13] using the EM (exact match) measure

The CS measure is considered 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 4, 5 and 6) with respect to the CS measure reveal the following: out of **24** comparisons, our approach is better in **17** cases, equal in **6** cases and slightly worse in **1** case (with a difference of 0.01). These results indicate the effectiveness of our proposal.

The running times of our algorithm have been emphasized in Table 2. The algorithm converges quite fast, in most cases in less than 2 s. However, for larger data sets (containing more sequences or having very long sequences), the algorithm needs more time to converge: for the considered data sets between

17 s and 7 min, depending on the data set. MAFFT and ClustalW are rapid algorithms as well. When measuring the computational time taken by these tools we notice that their running time does not change very much for larger data sets: we measured the running time for MAFFT for two of the evaluated data sets, one with 8 and one with 3 sequences, where the sequences have approximately equal lengths (first mitochondrial human DNA data set and data set 469 from oxbench_mdsa_all). The running times are very similar, being between 3 and 5 s, for both these data sets. However, we remark that our algorithm is somehow sensitive to the size of the input data: greater input data does not always mean more time, but there are cases in which this statement is true. For one of the data sets our algorithm needed 17 s to converge to the optimal alignment, but for the other it needed less than a second. We could therefore conclude that for smaller data sets our algorithm might be a better choice, from a computational time perspective and also from the point of view of the accuracy of its results.

The main approaches existing in the MSA literature (see Sect. 2.2) usually follow GA-based algorithm variations to solve the MSA problem. As opposed to the genetic algorithms solutions in the literature which do not alter the order of the sequences in the sequence set but randomly insert gaps in any of them by use of genetic operators in order to eventually improve the score of the candidate alignments, the proposed reinforcement learning approach tries to preserve the incremental pairwise nature of the major currently employed MSA tools (such as ClustalW, MAFFT) while exploring the search space in an effective reward-targeted manner. The results therefore obtained can be easily used for the construction of phylogenetic trees due to their progressive nature.

## 6 Conclusions and Further Work

In this paper we have approached, from a computational perspective, the multiple sequence alignment problem, using a reinforcement learning based approach and focusing on DNA sequences. To our knowledge, reinforcement learning has not been used in the literature to address this problem, so far.

The experimental evaluations have been conducted using real DNA data sets. The obtained results indicate the efficiency of our approach and prove that the RL based algorithm outperforms, in most cases, other techniques from the literature that approach the same problem. Moreover, another advantage of our algorithm is the low computational time, as the convergence is achieved in a matter of seconds or at most minutes, for the considered data sets.

We will investigate possible improvements of the model by using different reinforcement functions, adding new local search mechanisms to increase the model's performance, using function approximation methods for the $Q$-values, in case the state space becomes very large. In addition, we plan to investigate a second RL based model, which uses different definitions of the state and action spaces and of the reinforcement function. Regarding the biological perspective, we will consider a modification of the alignment algorithm so as to favor both global alignment and the finding of motifs that are locally similar in the sequences [11].

**Acknowledgment.** This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-II-RU-TE-2014-4-0082.

## References

1. Agarwal, P.: Alignment of multiple sequences using GA method. Int. J. Emerg. Technol. Comput. Appl. Sci. (IJETCAS) **13–177**, 412–421 (2013)
2. Carroll, H., Beckstead, W., O'Connor, T., Ebbert, M., Clement, M., Snell, Q., McClellan, D.: Dna reference alignment benchmarks based on teritary structure of encoded proteins. Bioinformatics **23**(19), 2648–2649 (2007)
3. Chao, L., Shuai, L.: The research on DNA multiple sequence alignment based on adaptive immune genetic algorithm. In: International Conference on Electronics and Optoelectronics (ICEOE), vol. 3, pp. V3–75–V3–78, July 2011
4. Chen, S.M., Lin, C.H.: Multiple DNA sequence aalignment based on genetic algorithms and divide-and-conquer techniques. Int. J. Appl. Sci. Eng. **3**, 89–100 (2005)
5. Chen, S.M., Lin, C.H.: Multiple DNA sequence alignment based on genetic simulated annealing techniques. Inf. Manag. Sci. **18**, 97–111 (2007)
6. Chen, Y., Pan, Y., Chen, L., Chen, J.: Partitioned optimization algorithms for multiple sequence alignment. In: Proceedings of the 20th International Conference on Advanced Information Networking and Applications, pp. 618–622 (2006)
7. Czibula, I., Bocicor, M., Czibula, G.: A software framework for solving combinatorial optimization tasks. Studia Universitatis "Babes-Bolyai", Informatica, **LVI**, 3–8 (2011). Proceedings of KEPT 2011, Special Issue
8. Dayan, P., Sejnowski, T.: TD($\lambda$) converges with probability 1. Mach. Learn. **14**, 295–301 (1994)
9. Eger, S.: Sequence alignment with arbitrary steps and further generalizations, with applications to alignments in linguistics. Inf. Sci. **237**, 287–304 (2013)
10. EMBL-EBI, The european bioinformatics institute. http://www.ebi.ac.uk/about
11. Gotoh, O.: An improved algorithm for matching biological sequences. J. Mol. Biol. **162**, 705–708 (1982)
12. Kanz, C., Aldebert, P., Althorpe, N., et al.: The EMBL nucleotide sequence database. Nucleic Acids Res. **36**, D29–D33 (2005)
13. Katoh, S.: MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol. Biol. Evol. **30**, 772–780 (2013)
14. Larkin, M., Blackshields, G., Brown, N., Chenna, R., McGettigan, P., McWilliam, H., Valentin, F., Wallace, I., Wilm, A., Lopez, R., Thompson, J., Gibson, T., Higgins, D.: ClustalW and clustalX version 2.0. Bioinformatics **23**(21), 2947–2948 (2007)
15. Lipman, D., Altschul, S., Kececioglu, J.: A tool for multiple sequence alignment. Proc. Natl. Acad. Sci. U.S.A. **86**, 4412–4415 (1989)
16. Mircea, I., Bocicor, M., Dîncu, A.: On reinforcement learning based multiple sequence alignment. Studia Universitatis "Babes-Bolyai", Informatica **LIX**, 50–65 (2014)
17. Nasser, S., Vert, G., Nicolescu, M., Murray, A.: Multiple sequence alignment using fuzzy logic. In: Proceedings of the IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, pp. 304–311 (2007)
18. Needleman, S., Wunsch, C.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. **48**(3), 443–453 (1970)

19. Nelwamondo, F.V., Golding, D., Marwala, T.: A dynamic programming approach to missing data estimation using neural networks. Inf. Sci. **237**, 49–58 (2013)
20. Nguyen, H., Yoshihara, I., Yamamori, K., Yasunaga, M.: Neural networks, adaptive optimization, and RNA secondary structure prediction. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, pp. 309–314 (2002)
21. Nizam, A., Shanmugham, B., Subburaya, K.: Self-organizing genetic algorithm for multiple sequence alignment. Glob. J. Comput. Sci. Technol. **11**(7) (2011)
22. Rasmussen, T., Krink, T.: Improved hidden Markov model training for multiple sequence alignment by a particle swarm optimization-evolutionary algorithm hybrid. BioSystems **72**, 5–17 (2003)
23. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
24. Thompson, J.D., Linard, B., Lecompte, O., Poch, O.: A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives. PLoS ONE **6**(3), e18093+ (2011)
25. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. Comput. Biol. **4**, 337–348 (1994)
26. Xiang, X., Zhang, D., Qin, J., Yuanyuan, F.: Ant colony with genetic algorithm based on planar graph for multiple sequence alignment. Inf. Technol. J. **9**(2), 274–281 (2010)