STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 2, 2014

ON REINFORCEMENT LEARNING BASED MULTIPLE SEQUENCE ALIGNMENT

IOAN-GABRIEL MIRCEA, MARIA-IULIANA BOCICOR, AND ANDRA DÎNCU

Abstract. Multiple alignment of biological sequences may reveal important functional, structural or evolutionary relationships between the sequences. Although very important from a biological perspective, the problem of multiple sequence alignment is quite challenging from a computational point of view, as it is NP-complete. Reinforcement learning is an approach to machine intelligence in which an adaptive system can learn to behave in a certain way by receiving punishments or rewards for its chosen actions. In this paper we investigate a reinforcement learning based model for the multiple sequence alignment problem, which combines a Q-learning algorithm with two variations of a sequence alignment algorithm and three different action selection policies. The model is experimentally evaluated on two data sets containing mitochondrial human DNA sequences from remains collected during several archeological excavations. The obtained results for each algorithmic combination are analysed and we provide comparisons of these results.

1. Introduction

In a bioinformatics context, sequence alignment refers to the process of arranging the primary sequences of DNA, RNA or protein to identify regions of similarity between them. This type of analysis has important applications in molecular biology, as similarities identified by an alignment of sequences may indicate functional, structural or evolutionary relationships between the sequences. Pairwise alignment involves only two sequences, but biologists are often interested in similarities between three or more. In such situations, multiple sequence alignment is performed to the goal of finding common bits from all the sequences in a given set.

Received by the editors: September 15, 2014.

2010 *Mathematics Subject Classification.* 68P15, 68T05.

1998 *CR Categories and Descriptors.* I.2.6[**Computing Methodologies**]: Artificial Intelligence – *Learning*; I.2.8[**Computing Methodologies**]: Problem Solving, Control Methods, and Search – *Heuristic methods*.

Key words and phrases. Bioinformatics, Multiple sequence alignment, Reinforcement learning.

Reinforcement learning (RL) [17] is an approach to machine intelligence in which an agent can learn to behave in a certain way by receiving punishments or rewards for its chosen actions. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the highest reward by trying them. The reinforcement learning algorithms selectively retain the outputs that maximize the received reward over time.

In this paper we aim to further investigate a reinforcement learning based approach to the multiple sequence alignment problem, approach that we have previously introduced in [1]. In this work we employ a variation of the previously used alignment algorithm and different action selection mechanisms in the RL process. Two human DNA data sets are used for the experimental evaluations and we provide analysis and comparisons of the obtained results.

The rest of the paper is organized as follows. Section 2 presents the biological multiple sequence alignment problem, in the context of bioarcheology, some methods existing in the literature for solving this problem, as well as the RL based approach that we have previously introduced [1]. A set of criteria and variations to the RL based approach, more specifically to the multiple alignment algorithm that was used and to the action selection mechanisms in the RL process are presented in Section 3. Experimental evaluations, analysis and comparisons of the all the algorithms are given in Section 4. Section 5 outlines our conclusions and further work.

2. Background

This section presents the multiple sequence alignment problem in bioinformatics and, more particularly, bioarcheology framework. Furthermore, we briefly review some fundamental aspects related to the RL based approach that we previously introduced for multiple sequence alignment. A short revision of some other techniques from the literature that address this problem is also provided in this section.

2.1. The Multiple Sequence Alignment (MSA) problem and its relevance in bioarcheology.

The field of bioarchaeology [6] is an interdisciplinary field of research that employs techniques and methods specific to biological research in order to aid archaeologists in obtaining additional information regarding human or animal remains located on the archaeological sites. The term of bioarchaeology was introduced in 1972 by the British archaeologist Grahame Clark in order to define the analysis conducted on the faunal remains from the prehistoric North Yorkshire site at Star Carr [6]. The European concept includes both research on human remains and on animal and vegetal remains and residues, whereas in the United States it has been defined,

mostly concomitantly, to have a stricter sense and only refer to human osteology in the work of Buikstra [2]. The main goal of the bioarchaeologist is to analyse the human remains found on an archaeological site in order to determine their sex, age and health as these indicators offer precious information about the social patterns in the society, culinary habits, etc.

A compelling application of multiple sequence alignment in bioarchaeology is presented in paper [16] which reveals the existence of domesticated turkeys on the south western region of the United States before the arrival of the Europeans. In order to achieve this task several mitochondrial DNA samples have been extracted from bones and coprolites and then compared against the consensus sequences by use of multiple sequence alignment as basis for the construction of the phylogenetic tree. Usually, the employment of MSA methods in bioarchaeology precedes the creation of phylogenetic trees which are later on used to genetically clusterize the material and then draw bioarcheologically relevant conclusions by analysing the data.

2.2. **Sequence to profile alignment.** The Multiple Sequence Alignment problem extends the classic dynamic programming pairwise alignment algorithms such as Needleman-Wunsch [13] to align more than two sequences. The problem that occurs in this respect is that while in the pairwise case the task of computing the alignment score is rather straightforward, in the multiple alignment case a sequence is usually aligned with a set of priorly aligned sequences.

In order to align a sequence to a set of already aligned sequences, the Needleman-Wunsch dynamic algorithm needs to be adapted to compute the score of each column in the sequence against all the corresponding symbols in the set of already aligned sequences. A profile of the partial alignment needs to be computed. The profile holds the frequencies of each symbol in the alphabet for each position in the set of pre-aligned sequences. This way, the pairwise comparison conducted in the Needleman-Wunsch algorithm can be substituted by a comparison between a symbol on a certain position in the sequence and the column corresponding to the same position in the profile.

2.3. **A reinforcement learning based approach for solving the MSA problem.** In [1] we introduced a reinforcement learning based technique for the multiple sequence alignment problem. Although in the above mentioned work we evaluated the approach on DNA sequences, the applicability of our method is more general and it can be used with other types of sequences as well (e.g. proteins).

From a computational point of view, the multiple sequence alignment problem was defined as the problem of finding the optimal permutation of input sequences, such that the score of the obtained alignment of the sequences

in the given order is maximal. The score is defined using the sum of pairs score method [11], which sums the substitution scores of all possible pairwise combinations of sequence characters over all columns of a multiple sequence alignment. The RL task associated to the MSA problem consists in training the agent to find a path from the initial to a final state having the maximum associated score. During the training step of the learning process the learning agent determines its optimal policy in the environment, i.e. the mapping from states to actions that maximizes the sum of the received rewards. The reward's definition is connected to the score computing method, such that by maximising the sum of rewards over time, the agent is actually maximising the obtained alignment's score. The equivalent action configuration is viewed as a permutation that gives the optimal alignment. For training the agent [1] a Q-learning approach was used [17] and a new action selection mechanism was defined in order to guide the exploration of the search space [1]. After the training step of the agent has been completed, the solution learned by the agent, which indicates the recovered alignment, is constructed starting from the initial state and following the Greedy mechanism. For more details about sequence to profile alignment, about the definitions of the state and action spaces, reward and transition functions or about the action selection mechanism, we refer the reader to [1].

2.4. **Literature review on MSA.** It has been shown that the MSA problem is NP-complete [19], therefore most algorithms that deal with this problem include different heuristics that produce quasi-optimal alignments, but are less computationally expensive.

An artificial intelligence method widely used for solving the MSA problem are Genetic algorithms (GA). Chen and Lin [3] tackle the DNA MSA problem using genetic algorithms in conjunction with divide-and-conquer techniques to determine the best cut points for DNA sequences. 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 [4]. A parallel hybrid genetic algorithm for the protein MSA problem, using new chromosome representation and corresponding new genetic operators and they use two kinds of local search heuristics was introduced by Nguyen *et al.* [14]. An improved GA with a research evolutionary computation system is presented by Zhang and Achawanantakun in [20].

Various other methods from the field of machine learning have been exploited and used to tackle the MSA problem. Chen et al. [5] present a partitioning approach combined with an ant colony optimization system, which solves the problem in three stages. Hidden Markov models are employed by

Rasmussen and Krink [15], in conjunction with a new particle swarm optimization based training method. Due to its tolerance of inexactness or errors in subsequence matching, fuzzy logic was used by Nasser et al. [12] for approximate matching of sequences.

3. Our study

In this section we aim to present several variations to our previously introduced RL approach, which will subsequently be experimented on, analysed and compared to our previously obtained results [1]. In [1] our RL technique used an extension of the Needleman-Wunsch algorithm [13] for detecting an optimal alignment of two or more sequences, which employs a linear gap penalisation process. In this paper we intend to use an *affine gap penalisation*, which favours both global alignment and minimises the interior gaps, thus leading towards also finding motifs that are locally similar in the sequences [9]. Furthermore, in [1], we introduced a new action selection mechanism for the RL process, which uses a one step look-ahead procedure (inspired from the ϵ-greedy policy) in order to guide the exploration of the search space. To investigate how the policy specifying the way in which a new action is chosen in each given state influences the accuracy of the final multiple alignment, we try two different action selection policies: ϵ-*Greedy* and *softmax*.

3.1. Needleman-Wunsch with affine gap penalty.

The Needleman-Wunsch algorithm [13] is a dynamic programming algorithm used for detecting the optimal alignment of two sequences (the alignment with the maximum score). The gap penalisation process used in the algorithm can be linear or affine. While the linear one considers the same penalty value for all the gaps, the affine version considers a *gap open penalty* given when opening a new sequence of gaps and a *gap extension penalty* that is given when adding gaps to an existing sequence. The cost for a gap in the affine gap penalty case is computed as $-d-(l-1)e$ where d is the gap open penalty and e is the gap extension penalty [8].

In the linear case there are three possible cases: when there was no gap (match or mismatch), when a gap was added in the first sequence and when a gap was added to the second sequence. For affine gap penalties, introducing a gap at the end of the current subsequence implies opening a gap sequence earlier in the alignment, thus all previous gap opening events must be considered.

For computing the optimal alignment, we denote three matrices:

- $G_d(i,j)$ best score up to position (i,j) and no gap at the end, diagonal direction

- $G_y(i,j)$ best score up to position (i,j) with a gap in the sequence y at position j, vertical direction
- $G_x(i,j)$ best score up to position (i,j) with a gap in the sequence x at position i, horizontal direction

For initializing the matrices we chose the following: $G_d(0,0) = 0, G_x(0,0) = -\infty, G_y = -\infty$. By following the rules from the previous equations, the rest of the elements from the first row and first column of G_d are $-\infty$. The rest of the initialization is done using the 0 in G_d and the gap open penalty first followed by gap extension penalties, thus the first row of G_y and the first column of G_x will be $-\infty$.

The value from each cell of the three matrices is computed by using the following equations:

$$G_y(i,j) = max \begin{cases} G_d(i-1,j) - |d| \\ G_y(i-1,j) - |e| \end{cases} \qquad G_x(i,j) = max \begin{cases} G_d(i,j-1) - |d| \\ G_x(i,j-1) - |e| \end{cases}$$

$$G_d(i,j) = max \begin{cases} G_d(i-1,j-1) + s(x_i, y_j) \\ G_y(i-1,j-1) + s(x_i, y_j) \\ G_x(i-1,j-1) + s(x_i, y_j) \end{cases}$$

where $s(x_i, y_j)$ is the match/mismatch score of x_i and y_j. After all values from G_d, G_y and G_x are computed, the global alignment can be determined by going backwards from the bottom-right cell with the highest score until the first colon or first row is reached by moving from a matrix to another depending on the names saved in the cells and the known directions. In this case, going from $(i-1, j-1)$ to (i,j) means there are no gaps added to the alignment, going from $(i, j-1)$ to (i,j) means there is a gap in the second sequence, while going from $(i-1, j)$ to (i,j) means there's a gap in the first sequence. Here, the difference from the linear case comes from the jumps performed between the matrices when reconstructing the alignment. Staying in the same matrix either implies no gaps (staying in G_d), or a gap penalized with the gap extension penalty, e, in the first or second sequence. The gap open penalty appears when a new gap sequence is open, thus in the backwards case, when a gap sequence finishes and when jumping from G_y or G_x to G_d. The optimal score of the global alignment determined is the maximum value found in the bottom-right cell of the three matrices.

In our case, the alignment is performed between a DNA sequence and a previously aligned profile. The algorithm acts the same, the difference being that by adding a gap into the profile, we add it to all its sequences at a certain position. The process of aligning a sequence to a profile was described in [1] and Section 2.2, the difference being the gap penalization method chosen for the Needleman-Wunsch algorithm.

An example was selected to illustrate the process of aligning a sequence to a profile by using the affine penalization method for Needleman-Wunsch algorithm. For this example, a simple sequence $s = AGT$ is aligned against the following profile (set of previously aligned sequences):

$$\begin{matrix} A & A & G & C & T \\ A & G & G & - & T \end{matrix}$$

For this, we consider the following scores: $match = 4$, $mismatch = -2$, $d = -3$, $e = -1$.

By following the previous described algorithm we obtain the following three matrices:

$$G_d = \begin{matrix} 0 & -\infty & -\infty & -\infty \\ -\infty & 4,d & -5,x & -6,x \\ -\infty & -2,y & 5,d & -1,x \\ -\infty & -6,y & 5,y & 3,d \\ -\infty & -6,y & -1,y & 4,d \\ -\infty & -8,y & -3,y & 6,y \end{matrix}; G_x = \begin{matrix} 0 & -3,d & -4,x & -5,x \\ -\infty & -\infty & 1,d & 0,x \\ -\infty & -\infty & -5,d & 2,d \\ -\infty & -\infty & -9,d & 2,d \\ -\infty & -\infty & -9,d & -4,d \\ -\infty & -\infty & -11,d & -6,d \end{matrix};$$

$$G_y = \begin{matrix} 0 & -\infty & -\infty & -\infty \\ -3,d & -\infty & -\infty & -\infty \\ -4,y & 1,d & -8,d & -9,d \\ -5,y & 0,y & 2,d & -4,d \\ -6,y & -1,y & 2,d & 0,d \\ -7,y & -2,y & 1,y & 1,d \end{matrix}$$

The optimal score can be found in the bottom-right corner of G_d, with value 6. By going backwards from this cell the following alignment is obtained:

$$\begin{matrix} A & A & G & C & T \\ A & G & G & - & T \\ A & G & - & - & T \end{matrix}$$

From this alignment the score for evaluating the alignment can be computed by using the sum of pairs metric [11].

3.2. **Action selection policies.** A very significant aspect of RL is a trade-off between *exploitation* and *exploration* [18]. The system aims to accumulate a lot of reward, therefore it will prefer the best experienced actions. However, it has to also try new actions, as these may lead to higher long term rewards. The rules or mechanisms that the system uses in order to make transitions among states during the learning process are the *action selection policies*. Several such policies exist in the literature.The *greedy* policy implies that the learning agent chooses the highest-valued action in each state. An agent using this mechanism only exploits current knowledge to maximize its reward, but

does not explore new states that could lead to higher long term rewards. A more effective method, which balances the exploration of new states with exploitation of current knowledge, is ϵ-Greedy [17]. It selects the greedy action with probability $1-\epsilon$ and, in order to explore the environment, with probability ϵ it chooses an action at random, uniformly, not taking into consideration the action value estimates. Therefore, the main drawback of ϵ-Greedy is that the worst action is as likely to be chosen as the second best one.The *softmax* action selection policy [17] was introduced to counter this disadvantage. A system using this mechanism will choose better actions more often. Actions are ranked according to their value estimates and each action is chosen with a probability computed using its value. The greedy action will still have the highest probability. The most common softmax method uses a Gibbs, or Boltzmann distribution, where the probability of choosing action a in state s is (for a Q-learning approach):

$$\frac{e^{Q(s,a)/\tau}}{\sum_a e^{Q(s,a)/\tau}} \tag{1}$$

where τ is a positive parameter called *temperature*, which specifies how random actions should be chosen. For high values of the temperature all actions will be almost equiprobable. As the temperature is reduced, the actions that have higher value estimates are more likely to be selected and in the limit, as $\tau \to 0$, the best action is always chosen, this meaning that the softmax policy becomes the same as the greedy policy.

4. Experimental evaluation

This section provides experimental evaluations of the RL approach, which uses the alignment algorithm presented in Subsection 3.1, as well as the different action selection policies: the one step look-ahead procedure introduced in [1] and the two policies presented in Subsection 3.2.

4.1. **Parameters setting.** For all our experiments we used a a software framework that we have previously introduced for solving combinatorial optimization problems using reinforcement learning techniques [7]. For all types of tests, we used the following values for the RL parameters: the discount factor for the future rewards is $\gamma = 0.9$; the learning rate is $\alpha = 0.8$; the number of training episodes is $5 \cdot 10^4$; for all three action selection policies, the policy parameter (ϵ - in the case of ϵ-Greedy and the one step look-ahead procedure and τ - in the case of softmax) is set to 0.8. Regarding the values used for the scoring matrix employed by the alignment algorithm, we considered the following values: -3 for the *gap penalty*, -2 for the gap extension penalty, -1 for the *mismatch penalty* and $+4$ for the *match score*. These values were

chosen after a literature study and so that no two values are equal. Also, it is important that the gap penalty is higher (in absolute value) than the gap extension penalty, because its aim is to encourage the extension of gaps rather than the introduction of new gap subsequences. This suggests that the gap cost function is not computed linearly, but considering all previous gap opening events.

The results obtained using the different alignment algorithms and various selection policies are compared by examining alignment quality, through a measure called *sum of pairs score* [11], the *ratio of matched columns* of the final alignment (computed as the number of columns containing symbols that match completely divided by the number of total columns of the alignment), and by the number of epochs and computational time needed by the RL algorithm to converge to the optimal solution. We mention that the higher the score and the number of matched columns, the better the algorithm.

We mention that all the experiments presented in this section were carried out on a PC with an Intel Core i7 Processor at 2.2 GHz, with 8 GB of RAM.

4.2. **Data sets - Mitochondrial human DNA.** We used two real-life data sets to test the performances of the different variations of our RL technique. Both sets contain human DNA sequences, obtained from the Babeş-Bolyai University Interdisciplinary Research Institute on Bio & Nano Sciences [10]. The first set is rather heterogenous, containing various samples of mitochondrial human DNA from remains collected during several archeological excavations. It is composed of 8 DNA sequences, with an average length of 365.75 nucleotides. The sequences from this set contain some positions which are marked with the character "N because it is not exactly known which specific nucleotide is found on that position. In contrast, the second set of sequences only contains sequences from a single archaeological site. There are only 6 DNA sequences in this case and the average sequence length is 362.83. A correct global alignment of the sequences in this second data set could provide compelling information to the biologists regarding the origins of the underlying population of the archeological site and the influence of other populations on the local gene pool.

4.3. **Comparative results.** We experimented on the original RL approach (the one using the Needleman-Wunsch algorithm [13] with linear gap penalty), as well as on the approach using affine gap penalty [9]. We used the action selection policies presented in Subsection 3.2. For each policy, the RL algorithm was run several times and the reported results are the ones retrieving the highest scores towards which the algorithm converges. In the case of the first data set the reported scores were obtained in several runs, although there were also times when the algorithm converged towards different scores. In the

	Score			MC			Epochs			Time (sec)		
	ϵ	**SM**	**LA**	ϵ	**SM**	**LA**	ϵ	**SM**	**LA**	ϵ	**SM**	**LA**
Linear	36645	36664	36664	0.839	0.842	0.842	13000	7100	13200	1334	497	1303
Affine	36650	36655	36653	0.842	0.842	0.842	5300	4800	2400	3649	3383	6610

TABLE 1. Results for the first data set, obtained using the Needleman-Wunsch alignment algorithm with both linear and affine gap penalty and different action selection policies in the RL process.

case of the second data set, however, the algorithm always converged towards the scores that are reported in the following.

Table 1 comparatively presents the obtained scores for the first data set, the ratio of matched columns, the epochs and computational time (in seconds) needed for convergence, for both linear and affine gap penalisation algorithms and all three action selection policies.

The first thing to note about the results of the first data set is that they fluctuate. In the linear case two of the action selection mechanisms lead to the highest obtained score, namely softmax and the look-ahead policy. The algorithm using the ϵ-greedy mechanisms converges, in the best case, to a lower score, but which is, however, only slightly lower than the other. Softmax should indeed (theoretically) converge towards better solutions 3.2, as it was designed to better exploit the already known information. The intelligent look-ahead policy was modeled so as to better guide the agent through the search space, therefore it also should (and does) lead to higher scores. An interesting thing to note is that, despite its design to better guide the agent through the search space, in the linear case, the intelligent selection policy needs the highest number of epochs to converge. We remark that the reported result is the one having the highest score, among all the runs of the algorithm and that in other runs that used this policy, the algorithm converged much faster, but towards a lower score. Softmax proves to be, in the linear case, the policy that leads to the fastest convergence, both in terms of epochs and computational time. In terms of number of epochs, the linear algorithm converges much later than the affine one, but this is not reflected in the running time. The time complexity of the affine algorithm is cubic, as opposed to the linear one, which is only quadratic. As at each step in an epoch the algorithm performs a new alignment, with a time complexity depending on the number of sequences at that step, it is expected that the total time of the affine version should increase much faster.

The Needleman-Wunsch algorithm with affine gap penalty [9] was first developed to overcome certain difficulties of the linear gap penalty algorithm and

	Score			MC			Epochs			Time (sec)		
	ϵ	**SM**	**LA**	ϵ	**SM**	**LA**	ϵ	**SM**	**LA**	ϵ	**SM**	**LA**
Linear	20337	20337	20337	0.911	0.911	0.911	2300	1200	500	101	91	76
Affine	20332	20332	20332	0.911	0.911	0.911	5200	4300	700	2042	1748	293

TABLE 2. Results for the second data set, obtained using the Needleman-Wunsch alignment algorithm with both linear and affine gap penalty and different action selection policies in the RL process.

was therefore expected to yield better results. However, we noticed that this is not always the case. There are permutations of the input sequences for which the affine algorithm obtaines higher scores, but there are also cases in which it does not. The RL algorithm converges to different scores (implicitly, different permutations of the sequences) for each of the action selection policies. Table 1 shows that the highest score was obtained when using the softmax policy, but this was however slighly lower than in the linear case. The only policy for which the affine algorithm reached a higher score was ϵ-greedy. Although the scores and the alignments for affine and linear are slightly different, we remark that in the affine case all of the alignments have 382 columns, out of which 322 are complete matches, therefore the ratio of matched columns is 0.842 in all three cases, similar to the linear case, for the best obtained scores.

The results obtained for the second data set are shown in Table 2. In this case, the results are more consistent and both Rl algorithms, using the linear and affine gap penalty converge towards the same permutation of sequences. The difference in scores is due to the different alignments obtained by the two algorithms. As mentioned for the first data set, the affine algorithm does not always obtain higher score alignments, compared to the linear one and this is an example of such a case. All three action selection policies lead to the optimal result, the difference residing in the number of epochs and, implicitly, the needed computational time for convergence. In this case the intelligent look-ahead action selection policy leads to the faster convergence, as it theoretically should, because it was designed to efficiently guide the agent through the search space by choosing at a given step in an epoch, with a certain probability, the alignment with the highest score. As for the first data set, softmax converges faster than ϵ-Greedy.

We observe that even though the number of epochs is similar, the computational time needed for convergence is significantly higher in the affine case. This happens because of the affine gap penalisation process used during the alignment, which leads to the alignment algorithm running much slower than the one using linear gap penalty. As mentioned for the first data set, the time

complexity for the affine case is cubic, compared to the linear case, which is quadratic and in the computational experiments, for a fixed permutation, the affine algorithm runs almost three times slower. Cumulatively, when considering the number of times the algorithm is run in one epoch and then the number of epochs needed for convergence, we notice a considerable increase in the computational time.

Comparatively, the two methods (RL using Needleman-Wunsch with linear gap penalty and RL using Needleman-Wunsch with affine gap penalty) are quite similar from the perspective of solution quality. We observe similar performances regarding the different selection mechanisms as well. Figure 1 illustrates the scores obtained for the alignment of the sequences from the second data set during the training process. The left-hand side of the figure illustrates these scores in relation to the number of epochs, while the right-hand side shows them relative to the computational time needed for convergence. Each plot depicts the comparative scores obtained using the alignment algorithms with both gap penalties. The similarities and dissimilarities in convergence speed, when regarding the epochs, respectively the computational time can be clearly seen in these plots. We notice that convergence is achieved after a higher number of epochs for the affine algorithm and it also needs considerably more time to reach the optimal solution.

4.4. **Discussion.** In this paper we have addressed the problem of aligning multiple DNA sequences and we have experimented with a reinforcement learning based approach, using various types of action selection policies and two different alignment algorithms. The original model is a Q-learning algorithm combined with an extension to the Needleman-Wunsch algorithm [13] for multiple alignment and using an intelligent look-ahead action selection mechanism [1]. Several variations to the original model were investigated: first, two other action selection policies were employed: ϵ-greedy and softmax (Subsection 3.2), and second the multiple alignment algorithm was modified to use an affine gap penalisation process, which both favours global alignment and minimises the interior gaps. This second method was used in conjunction with all three action selection mechanisms.

The obtained results show that all three action selection policies lead to accurate results. From Tables 1 and 2 we notice that for both used data sets the scores of the obtained alignments are equal (and optimal) in most cases, or very similar in other cases, regardless of the selection mechanism. Concerning the number of epochs we observe that softmax always converges faster than ϵ-greedy. The intelligent look-ahead mechanism, which was developed to guide the search through the search space should converge faster than both, but this only happens in the case of the second data set.

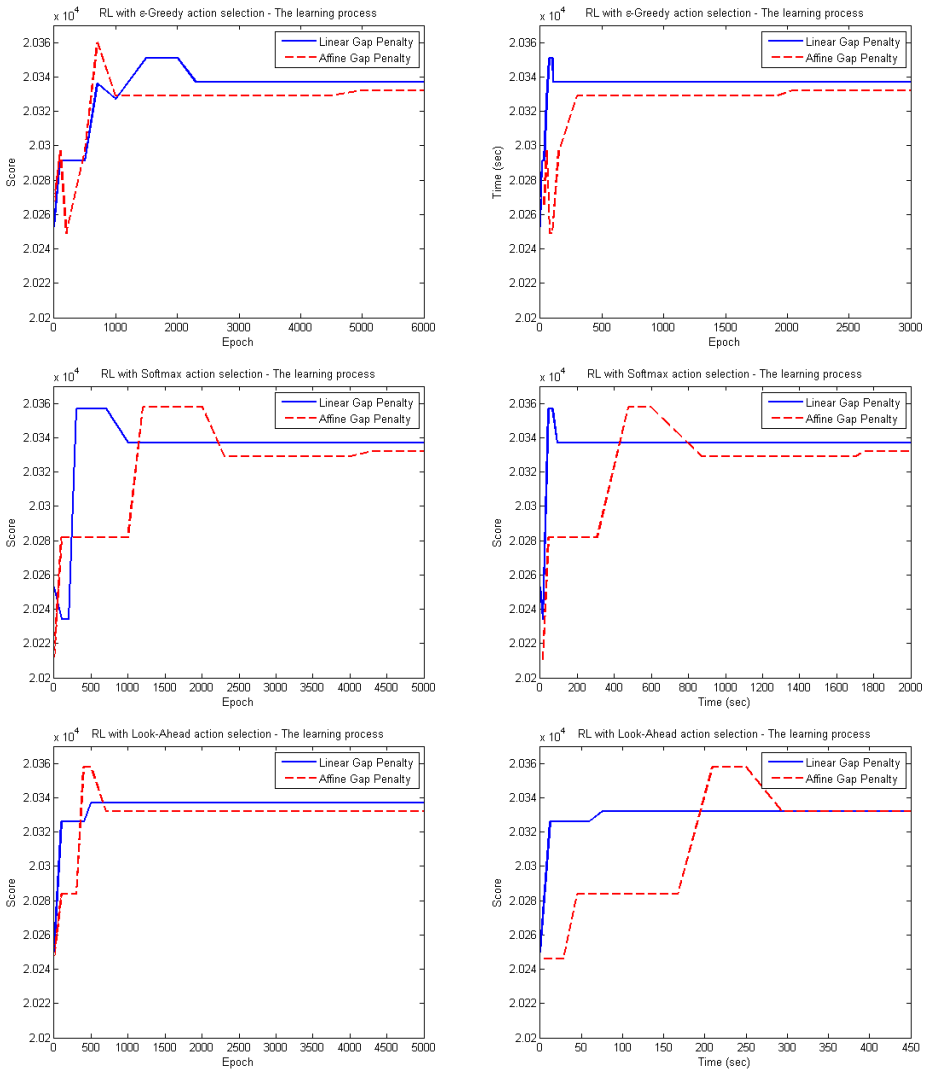

FIGURE 1. The learning process with the three action selection mechanisms, relative to the number of training epochs and to the computational time.

Our RL algorithm reaches the optimal or close-to optimal sequence alignments both when it uses the linear and affine gap penalty in the Needleman-Wunsch alignment algorithm. However, although the accuracies of the results are comparable, the main difference resides in the computational time and

this is true for both data sets. The time complexity of the affine algorithm is $O(n^3)$, while the one of the linear algorithm is $O(n^2)$. In practice, the alignment process for the linear algorithm is almost three times faster than that for the affine. The scores obtained for the first data set are slightly different, for each selection policy, both in the linear and the affine case. Although the alignments do not have significant differences, we believe that these variations in scores are due to the errors that exist in this data set: three of its sequences contain several positions marked with "N", because the specific nucleotides on those positions could not be determined. The second data set does not contain such error and thus the results are consistent, as can be seen in Table 2.

5. Conclusions and further work

The present study aimed to investigate several variations to a reinforcement learning based approach for the DNA sequence alignment problem. If the original RL based solution [1] employed the Needleman-Wunsch algorithm [13] with a linear gap penalisation process, in this paper we used the same algorithm, but with affine gap penalty. Furthermore, we experimented with two other action selection mechanisms in the RL process. The different combinations of gap penalty and action selection policies led to several algorithms, which were experimentally evaluated on two data sets containing real human DNA sequences. Through the obtained results the algorithms were analysed and compared.

The tests showed that both algorithms (with linear and affine gap penalty) obtain correct alignments, with small differences. Although the affine version is theoretically better (should obtain more accurate alignments), in our experiments the scores it obtained are slightly lower than those obtained by the linear algorithm (this implying slightly poorer sequence alignments), even if in some cases the permutations of sequences are exactly the same. Another difference worth mentioning is the computational time: in all cases the linear algorithm converges to the solution considerably faster than the affine algorithm, as it was expected, considering the time complexities of both algorithms. In conclusion, we can assert that for the considered data sets, the RL algorithm using the linear gap penalty alignment algorithm outperforms the one using the affine gap penalty process both in terms of accuracy and in terms of speed.

We plan to extend the evaluation of the RL based algorithms with both gap penalty processes for other data sets, to further develop the analysis. Also, we will use different parameters both for the alignemt algorithms and for the RL approach. We will investigate possible improvements of these models by

adding various local search mechanisms or combining the softmax policy with the intelligent action selection procedure.

Acknowledgments

We wish to thank our thesis supervisor, Professor Gabriela Czibula for the guidance, ideas and suggestions for this paper. Also, we gratefully acknowledge the assistance received from the Babeş-Bolyai University Interdisciplinary Research Institute on Bio & Nano Sciences [10] for providing several DNA data sets used in the experimental part of the paper, as well as for the feedback regarding the obtained computational results. Special thanks are due to Professor Beatrice Kelemen and her research team.

References

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

[2] Jane E. Buikstra. Biocultural dimensions of archeological study: A regional perspective. in biocultural adaptation in prehistoric america. In *Proceedings of the Southern Anthropological Society 11*, pages 67–84, Athens, GA, 1977. Univ. of Georgia Press.

[3] Shyi-Ming Chen and Chung-Hui Lin. Multiple DNA Sequence Aalignment Based on Genetic Algorithms and Divide-and-Conquer Techniques. *International Journal of Applied Science and Engineering*, 3:89–100, 2005.

[4] Shyi-Ming Chen and Chung-Hui Lin. Multiple DNA Sequence Alignment Based on Genetic Simulated Annealing Techniques. *Information and Management Sciences*, 18:97–111, 2007.

[5] Y. Chen, Y. Pan, L. Chen, and J. Chen. Partitioned optimization algorithms for multiple sequence alignment. *Proceedings of the 20th International Conference on Advanced Information Networking and Applications*, pages 618–622, 2006.

[6] J. G. D. Clark. *Star Carr: A case study in bioarchaeology.* Reading, MA: Addison-Wesley, 1972.

[7] I.G. Czibula, M.I. Bocicor, and G. Czibula. A software framework for solving combinatorial optimization tasks. *Studia Universitatis "Babes-Bolyai", Informatica, Proc. of KEPT 2011, Special IssueSpecial Issue*, LVI:3–8, 2011.

[8] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis.* Cambridge University Press, 1998.

[9] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of molecular biology*, 162:705–708, 1982.

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

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

[12] S. Nasser, G.L. Vert, M. Nicolescu, and A. Murray. Multiple sequence alignment using fuzzy logic. *Proceedings IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*, pages 304–311, 2007.

[13] S. 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*, 48(3):443–453, 1970.

[14] H.D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga. Neural networks, adaptive optimization, and RNA secondary structure prediction. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, pages 309–314, 2002.

[15] T.K. Rasmussen and T Krink. Improved hidden markov model training for multiple sequence alignment by a particle swarm optimization-evolutionary algorithm hybrid. *BioSystems*, 72:5–17, 2003.

[16] Camilla F. Speller, Brian M. Kemp, Scott D. Wyatt, Cara Monroe, William D. Lipe, Ursula M. Arndt, and Dongya Y. Yang. Ancient mitochondrial DNA analysis reveals complexity of indigenous North American turkey domestication. *Proceedings of the National Academy of Sciences*, 107(7):2807–2812, February 2010.

[17] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[18] Sebastian B. Thrun. The role of exploration in learning control. In D.A. White and D.A. Sofge, editors, *Handbook of intelligent control: Neural, fuzzy and adaptive approaches*. Van Nostrand Reinhold, New York, NY, 1992.

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

[20] Y. Zhang and R. Achawanantakun. An Improved Genetic Algorithm for Multiple Sequence Alignment. *Project Report of CSE848*, pages 1–7, 2010.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania

E-mail address: mircea@cs.ubbcluj.ro

E-mail address: iuliana@cs.ubbcluj.ro

E-mail address: dasi0581@scs.ubbcluj.ro