Exploring Reinforcement Learning Methods for Multiple Sequence Alignment: A Brief Review

Chaimaa Gaad[1*], *Mohamed-Amine Chadi*[2], *Mohamed Sraitih*[3], and Ahmed Aamouche[1]

[1] *LISA Laboratory, National School of Applied Sciences*, *University of Cadi Ayyad,* Marrakech, Morocco
[2] *LISI Laboratory*, *Computer science department, Faculty of Sciences Semlalia*, *University of Cadi Ayyad,* Marrakech, Morocco
[3] *MSC Laboratory, National school of applied sciences, University of Cadi Ayyad,* Marrakech, Morocco

Abstract. Multiple sequence alignment (MSA) plays a vital role in uncovering similarities among biological sequences such as DNA, RNA, or proteins, providing valuable information about their structural, functional, and evolutionary relationships. However, MSA is a computationally challenging problem, with complexity growing exponentially as the number and length of sequences increase. Currently, standard MSA tools like ClustalW, T-Coffee, and MAFFT, which are based on heuristic algorithms, are widely used but still face many challenges due to the combinatorial explosion. Recent advancements in MSA algorithms have employed reinforcement learning (RL), particularly deep reinforcement learning (DRL), and demonstrated optimized execution time and accuracy with promising results. This is because deep reinforcement learning algorithms update their search policies using gradient descent, instead of exploring the entire solution space making it significantly faster and efficient. In this article, we provide an overview of the recent historical advancements in MSA algorithms, highlighting RL models used to tackle the MSA problem and main challenges and opportunities in this regard.

Keywords: Multiple sequence alignment, Reinforcement learning, Computational complexity, Bioinformatics, Brief review.

1 Introduction

Multiple sequence alignment (MSA) is a crucial problem in bioinformatics that involves aligning multiple biological sequences, such as RNA, DNA, or protein sequences, to identify their evolutionary relation and functional similarity. In essence, MSA aims to organize sequences in a way that maximizes the similarity between the corresponding positions of the sequences. This is accomplished by including spaces (gaps) of various lengths within the sequences, aligning homologous positions in a manner similar to aligning beads of the same color in an abacus (Figure 1). Evolutionwise, these gaps symbolize indels (i.e., insertions and deletions) that are believed to have happened during the evolutionary process from a shared ancestor [1]. MSA is widely used in various applications in bioinformatics, including protein structure prediction, phylogenetic tree construction, and functional element identification in genomes. A recent paper [2] highlights the widespread use of multiple sequence alignment (MSA) in the field of biology. One of the most commonly used MSA methods, according to the study, is ClustalW [3], it is ranked as the 10th most cited scientific paper of all time. This demonstrates the significant impact that MSA has had on a wide range of in-silico analysis.

Since its inception, MSA has been tackled using a variety of methods, including probabilistic models [4], dynamic programming (DP) [5] such as the Needleman-Wunsch [6] and Smith-Waterman [7]. Dynamic programming is also used in the progressive alignment (PA) algorithm, which was first described by Hogeweg and Hesper [8]. Progressive alignment is one of the most used heuristics, A guide tree is constructed at the beginning of the algorithm to determine the order in which the input sequences will be incorporated into the final alignment. At each step, the algorithm performs a pairwise alignment between two sequences, profiles, or a combination of both, using a global dynamic programming algorithm. This combination of a tree based approach and a global pairwise alignment is the foundation of many popular MSA methods, including T-Coffee [9], and ProbCons [10]. These methods can be further improved by using iterative strategies [11], which involves repeating the process of estimating the guide tree and aligning sequences until they converge, as performed in methods such as MAFFT [12], MUSCLE [13], and Clustal Omega . These methods have been successful in providing accurate and fast solutions to the MSA problem, especially for small and moderate-sized datasets. However, as the size and complexity of biological datasets continue to increase, there is a growing need for more advanced and efficient MSA methods that can handle large-scale datasets and complex evolutionary relationships. The development of MSA as a valuable modeling tool has required

* Corresponding author: chaimaa.gaad@ced.uca.ma

overcoming a set of complex computational and biological challenges. Due to its NP- complete (non--

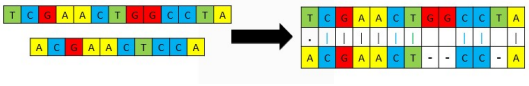

Fig. 1. Representation of 2 sequences of nucleotides (left), and the optimal alignment of these 2 sequences (right). Where (.) represents a substitution, (|) represents a match and ( ) represents a gap or an indel.

-deterministic polynomial time) nature [15], which means that finding an optimal solution for large-scale sequence datasets is difficult and often requires exponential time, the calculation of a precise MSA has been a long-standing issue in the field, resulting in the creation of over 100 distinct methods in the past thirty years [16].

Multiple sequence alignment (MSA) metrics are crucial in assessing the accuracy and quality of an alignment produced by a given method or software. The most commonly used metrics to evaluate the MSA quality are the sum-of-pairs (SP-score) and the column scores (CS). The SP score is a measure of the percentage of correctly aligned residue pairs in the alignment that indicates how well the concerned program is able to align at least some of the sequences [17]. On the other hand, the column score measures the percentage of the columns aligned correctly in the alignment, which assesses the program's ability to align all the sequences accurately. The calculation of column score (CS) involves taking the ratio between the number of columns that perfectly match (i.e., the exact match EM) across the entire alignment and the total length of the alignment (i.e., alignment's length AL) (CS = EM / AL). Both scores are valuable in determining the overall performance of the program in generating a reliable multiple sequence alignment [18].

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

Recently, reinforcement learning (RL), a branch of artificial intelligence (AI), that involves training an agent to make decisions in an environment by maximizing a reward signal, has been proposed as a new approach for solving the MSA problem. RL-based MSA methods aim to align sequences by learning a policy that guides the agent to make optimal alignments based on a reward signal that reflects the quality of the alignment. The key advantage of RL-based MSA methods is that they can effectively capture the complex relationships between sequences and handle different datasets by leveraging the generalization and representation learning capabilities of deep neural networks. The following article aims to provide a brief and comprehensive understanding of the application of RL multiple sequence alignment (MSA) problem. To achieve this goal, the article is structured into several sections. The first section is the introduction, which provides a comprehensive overview of MSA, the traditional tools to solve its problem and its relevance to bioinformatics. The next section will define RL, its components, as well as its types to give readers a decent understanding of the fundamental concepts related to the topic and will delve into the various types of RL used in the state-of-the-art research on MSA, including Q-learning and A3C. The third section is a literature survey, which will provide an overview on the applications of RL based approaches for MSA problem, a comparison in term of performance for these applications and a discussion. Finally, the conclusion section will summarize the findings and provide insights into future directions of research in this field.

Table 1. Notation used in the SP-Score function.

Notation	Meaning
n	is the number of sequences in the alignment
N	is the number of columns in the alignment
$\boldsymbol{c}_i$, $\boldsymbol{i}\epsilon\{\boldsymbol{1}, \dots, \boldsymbol{N}\}$	is the column i from the alignment
$\boldsymbol{c}_j$, $\boldsymbol{j}\epsilon\{\boldsymbol{1}, \dots, \boldsymbol{N}\}$	is the jth character from the i^{th} column
$s(c_i^j c_i^k)$	is a measure that indicates the comparing score between characters c_i^j and c_i^k

2 Reinforcement learning methods

2.1 Key concepts

Reinforcement Learning is a type of machine learning (ML), that focuses on teaching an agent to make decisions by taking actions in an environment to maximize a reward signal (Figure 2) [19]. Deep RL (DRL) when coupled with neural networks, we get. The goal of RL is to discover the best policy, or a mapping from states to actions, that maximizes the expected reward over time.

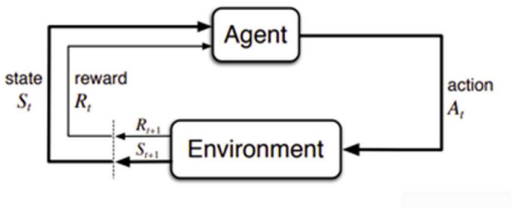

Fig. 2. The agent-environment interaction in RL.

Reinforcement Learning is based on several key concepts, including the agent, environment, state, action, reward, and policy (Table 2). There are several different types of RL algorithms, including value-based such as Q-learning algorithms, which was first introduced in 1992 by Watkins and Dayan [20], policy-based algorithms such as REINFORCE [21], and actor-critic algorithms that combines the two set of information treatment such as A3C (Asynchronous Advantage Actor-Critic) [22]. These algorithms differ in how they approach the problem of finding an optimal policy, but all rely on the same basic concepts. RL offers

several advantages for MSA, such as (1) The ability to learn from experience: RL algorithms can learn from experience, through trials, which means that they can improve the performance over time without the need for explicit supervision. (2) Flexibility: RL algorithms are flexible and can be applied to a wide range of problems, including problems in robotics, control, and optimization (3) Adaptability: RL algorithms are able to adapt to changes in the environment [23], which is important in MSA where the data and the conditions of the environment can change over time, since we are dealing with a text-based data. (4) Robustness: RL algorithms are robust and can handle noisy and incomplete information, which is common in MSA. (5) Optimization: RL algorithms can be used to optimize the parameters of MSA algorithms, leading to improved performance.

2.2 RL algorithms used for MSA

Multiple sequence alignment (MSA) is a central problem in bioinformatics that has attracted a lot of attention in recent years, especially, with the challenge of aligning multiple sequences with different lengths.

In recent years, there has been limited research exploring the application of RL algorithms into MSA. RL-based MSA methods view the MSA problem as a sequential decision-making process, in which the goal is to maximize a reward function that evaluates the quality of the alignment. There are several RL-based MSA methods that have been proposed in recent years, each with its own approach to defining the environment, state, action, reward, and policy. Some of these methods have been based on the value-based algorithms, such as Q-learning and deep-Q-learning (DQN), while others are based on actor-critic algorithms, such as A3C.

2.2.1 Q-learning and DQN

Q-learning is among the very first models introduced in RL. It is a popular RL algorithm used to solve problems where the agent must make a sequence of decisions in an uncertain, dynamic environment to maximize a long-term reward. The key idea behind Q-learning is to learn an action-value function, also called a Q-function, which assigns a value to each action in a given state. The Q-function represents the expected cumulative reward that the agent will receive if it takes a particular action in a specific state and then follows the policy thereafter. During training, the agent uses the Q-function to choose its actions based on the current state and updates the Q-values of the action-state pairs based on the rewards obtained and the estimated value of the next state-action pair. This update rule is known as the Bellman equation [24], which is based on the principle of optimality stating that the optimal policy must satisfy the property where the value of the state is equal to the expected return for the best action. DQN (Deep Q-Network) is a RL algorithm that combines Q-learning with deep neural networks. Popularized in 2013 and has been used in various applications, including playing video games at superhuman level, robotic control [25], and natural language processing (NLP) [26].

Table 2. Key components of RL algorithm.

Element	Style
Agent	An agent is the decision-making entity in RL. It is responsible for taking actions in the environment and receiving rewards for those actions
Environment	The environment is the setting in which the agent operates. It can be a real-world environment, such as a stock market or a video game, or a simulated environment created in a laboratory. The environment provides information about the state of the world and gives the agent the opportunity to take actions that affect the world.
State	A state is a description of the environment at a given time. i.g, in a video game, the state might include information about the position of the player, the position of enemies, and the state of the game world
Action	An action is a decision that the agent makes in the environment. i.g, in a video game, an action might be to move left, right, up, or down
Reward	A reward is a signal that provides feedback to the agent about the quality of its actions, it can be a negative one (penalty). For instance, in a video game, the agent would possibly receive a reward for collecting coins or a penalty for losing a life. The agent uses the reward signal to update its policy and improve its performance over time
Policy	A policy is a mapping that goes from states to actions. It is the strategy that the agent learns through trial and error in the environment. The goal of RL is to discover the best policy that maximizes the expected reward over time

2.2.2 Asynchronous Advantage Actor-Critic

Asynchronous Advantage Actor-Critic (A3C) is a deep RL algorithm that is designed to achieve more efficient and effective learning. It combines the actor-critic approach with asynchronous parallelization. The actor-critic approach is a RL technique that consists of two components: the actor and the critic. The actor selects actions based on the current policy, while the critic evaluates the value of the current state. The actor uses feedback from the critic to improve its policy. In A3C, a neural network is used to represent both the actor and the critic. A3C also utilizes asynchronous parallelization, which enables multiple instances of the environment to run in parallel. Each instance updates the shared neural network parameters asynchronously, allowing the algorithm to take advantage of multiple cores or machines. This can greatly speed up the learning process and make it more efficient. By combining the actor-critic approach with asynchronous parallelization. A3C can improve learning efficiency and speed. This makes it a popular choice for applications in which a large amount of training data and computational power are needed.

3 RL applications: MSA insights

3.1 Applications from the literature

Tackling the MSA problem using RL was first introduced in 2016 [27]. In this paper, using Q-learning, the RL task is formulated as a search problem sequence and is represented as a set of multi-dimensional arrays, each array is a sequence of nucleotides {C, T, A, G} and (-) represents the gap, where the goal is to find the optimal path through a state space that leads to the optimal solution (i.e., the optimal alignment of the input sequences). The state space consists of all possible permutations of the input sequences, where each permutation represents a possible order for aligning the sequences. The RL agent's goal here is to find the permutation that leads to the optimal alignment of the input sequences. The agent's actions consist of selecting one of the input sequences at each step. The agent can only select each sequence once, and it must select all the sequences exactly once, in an order that leads to the optimal alignment. The reward function is defined as the score of the alignment associated with the selected permutation of the input sequences, the score is determined by comparing the aligned sequences to a reference alignment and summing the scores of the individual matches, mismatches, and gaps. These steps are done by using a matrix that is a representation of each nucleotide and gap locus.

In another paper [28], the A3C algorithm was employed to address the problem of speed in MSA. The scoring scheme used to optimize the SP score is the linear gap penalty approach, with a score of (-1, +1, -1) for (gap, match, mismatch). In this paper, authors defined the state as an n×b matrix (n is the number of sequences of maximum length L) called the game board, $b \geq L$, and each cell contains either a gap or a nucleotide. The agent's role is to push nucleotides to the left or to the right to alter the alignment. The reward function assigns a real value to each possible (state, action) pair based on the SP-score of the resulting alignment. The goal state is not known ahead of time, so the agent is allowed to perform nSteps (which is a fixed number of steps) before the episode is stopped. The state with the maximum SP-score is labeled the ad hoc goal state, and actions taken after visiting that state are discarded. The policy followed by the agent consists of selecting an action based on the current state. The value of a (state, action) pair corresponds to its long-term expected discounted reward. The study approximates the value for each of the possible actions.

Additionally, the work presented in [29] proposed another approach based on the Markov decision process (MDP) framework with a state space consisting of all possible states, and an action space consisting of n actions, one for each sequence. The reward function is defined based on the SP-score, and the agent receives a large negative reward (-∞) if it chooses an action that is not distinct. The ultimate goal is to find an optimal policy (to maximize the expected reward). The model uses a neural network (NN) architecture based on Long Short-Term Memory (LSTM) to approximate the Q-values. The LSTM network has three hidden layers, each with 40 neurons, and a dropout layer to prevent overfitting. The input to the network is the state information needed to compute the SP-score, and the output is a list of chosen sequences. The model uses the experience replay method to prevent overfitting and improve stability, and the actor-critic algorithm for faster convergence, in this context, the actor refers to a policy that needs to select an action, while the critic is represented by the Q-function, which provides feedback to the actor regarding the optimality of its chosen action. The training loop of the model involves initializing the replay memory and the Q-network with random weights, selecting a random action with a small probability (to ensure sufficient exploration), observing the next reward and state, and storing the transition in the replay memory. The model then trains the network using mini-batch sampling and gradient descent, updating the weights with each iteration.

In a more recent study, the authors proposed an approach based on DQN [30], that is designed to guide progressive alignment using a combination of a new profile algorithm and negative feedback policy. Here the environment is the MSA task and the agent is responsible for computing the profile-sequence alignment, selecting the next sequence to align, and updating the state. The state in this RL environment is represented by a matrix T, which is padded with zeros (0) to a fixed size, the non-zero entries represent the sequences that have already been aligned, and the order of the entries must follow a specific rule to ensure that all non-zero numbers are adjacent. The action the agent takes is selecting the next sequence to align, and the reward is determined by the quality of the alignment resulting from the chosen action. The reward function also includes a negative reward for choosing a sequence that has already been aligned. The negative feedback policy role is to randomly select an action with a certain probability or uses the DQN model to predict the action value for each possible action. The DQN model is trained using a replay memory that saves all the invalid transitions that aligns the current profile with the new sequence selected by the agent and updates the current state. The transition function is a quad-tuple consisting of (the current state, the selected action, the resulting next state, and the associated reward).

3.2 Benchmarking

To conduct our benchmark, we used various well-known datasets that have been used in previous state-of-the-art studies. the first, contains DNA sequences from three distinct species: lemur, gorilla, and mouse. The average length of these sequences is 93 nucleotides. Similarly, the rat, lemur, and opossum dataset, which was used in the same study [31], consists of the third exon sequences of the beta globin gene and has an average sequence length of 129 nucleotides. We also included two datasets from the oxbench mdsa all [32] database: dataset 429, which has an average sequence length of 171 nucleotides, and dataset 469, which has an average sequence length of 332 nucleotides. Finally, we used two datasets from the European Molecular Biology Laboratory (EMBL) Nucleotide Sequence Database [33]: one dataset containing 10 sequences from the Hepatitis C virus (sutype 1b), with an average sequence

Table 3. Comprehensive comparison of the SP-Score metric.

Dataset	Number of sequences	Average length	Q-Learning [27]	DQN [29]	DQN [30]	ClustalW [27]	MAFFT [27]
Lemur, gorilla, mouse, dataset	3	93	345	345	**348**	345	345
429 from oxbench_mdsa_msa	12	171	8668	**10218**	8784	9575	10218
Papio Anubis	5	1093	18719	18860	**18968**	18827	18860
469 from oxbench_mdsa_all	3	332	565	565	**587**	464	549
Rat, lemur, opssum dataset	3	129	**486**	**486**	482	480	471
Hepatitis C virus	10	211.9	**18627**	**18627**	**18627**	**18627**	**18627**

length of 211 nucleotides, and another dataset containing 5 sequences from the Papio (Anubis olive baboon) organism, with an average sequence length of 1093 nucleotides. Note: In Table 3 below, we included all papers presented previously that used RL for solving the MSA problem. However, for [28], where authors used the A3C algorithm, the metric results were not mentioned, therefore, we did not include it in Table 3.

3.3 Discussion: challenges and opportunities

Multiple sequence alignment is a mandatory and fundamental technique in bioinformatics that aims to evaluate, distinguish, and identify similarities and differences between sequences. This knowledge is critical for scientists to understand the structure and function of biological molecules and can help with drug development, evolutionary biology, and comparative genomics studies.

Several approaches have been used such as probabilistic models [4], dynamic programming (DP), and progressive alignment. These classical techniques have been widely used for decades in the research area and have successfully provided accurate and fast solutions to perform the MSA, especially for limited datasets. However, these techniques are limited to a certain amount of data and a certain level of complexity, which elevates the need for more advanced methods suitable for dealing with large-scale data and intricate evolutionary interactions. In recent years, reinforcement learning emerged as a promising technique to elevate this issue. Nevertheless, as previously mentioned, only four studies suggested the RL for solving the MSA so far. This may appear insufficient for a full review, even though, reviewing these papers can provide insights into how these methods perform and if there is room for improvements. Each one of the reviewed papers used a RL model to perform the MSA by introducing a different formalization of the problem, yet to achieve the same goal.

As mentioned previously, RL is based on four core concepts (state, action, environment, reward). However, each author used these concepts differently by making the agent choose the sequence to use for the next round to extract the profile [29] and [30] or to find the path to the highest score linked to a certain alignment as in [27]. Or by taking actions on the sequence itself by moving the nucleotides or inserting gaps [28]. Therefore, the performances are quite different from one approach to the next for the same dataset, whereas, for others, all the approaches achieved the same results. Table 1 illustrates the performance of the SP score achieved by each approach evaluated on different datasets. The authors in this paper [30] achieved the highest SP score in terms of the majority of the experimented datasets by arising 0.57%, 0.87%, and 3.89% for Papio Anubis, Lemur, gorilla, mouse, and "629 from oxbench_mdsa" _all datasets, respectively. On the other hand, [28] and [29] achieved lower performance results for the same previous two datasets. Additionally, for the "429 from the oxbench_mdsa_msa" dataset, achieved the highest SP score by a rise of 16.32% in comparison with other approaches, keeping in mind that this is the largest dataset among the others. This could be because the authors implemented a strategy that makes the agent receive a large negative reward (-∞) if it chooses an action that is not distinct in finding the optimal policy (to maximize the expected reward). As well as a high performance, similar to [28] for the Rat, lemur, opssum dataset by a rise of 0.82%, which is the highest value it could be achieved specifically for this dataset as [30] didn't achieve better than that. Finally, for Hepatitis C virus dataset, all techniques achieved the same top SP score. This may be explained by the fact that for this specific dataset, all approaches were able to obtain the same ideal alignment, which reflects the greatest SP score. Furthermore, all these approaches have been compared to the classical methods, which illustrates that the novel approaches achieved better and equal performance in some cases and a lower SP score in others. However, this could be explained by the fact that the agent could find the optimal MSA by reaching the highest SP score. In other words, the RL approaches can explore the search space in an efficient reward-targeted way, which can provide a higher chance to find the optimal MSA for a given dataset.

Even with these performances, there are still several challenges that need to be addressed to boost the efficiency of reinforcement learning in MSA. One of the main challenges is having a high generalization ability, as the sequence size and number vary in different datasets. This issue can be investigated by using a diverse and representative training set with different model parameters. In addition to efficient reward

functions. However, as the sequence scale is large, it is more complex, time and memory-consuming, which leads to the exponentially growing size of the state space. This needs to be solved by investigating transfer learning, where the agent is trained on a reasonable number of datasets to initialize networks for larger-scale sequences. Additionally, using the profiling algorithm, which reduces the computational complexity of the iterative process, improves speed without compromising accuracy.

4 Conclusion

In conclusion, the four articles presented different RL-based approaches for solving the MSA problem in bioinformatics. While each approach had its own strengths and limitations, they all showed promising results in improving the accuracy and efficiency of MSA. One key challenge for future research is to address the issue of generalization to longer and more numerous sequences, while using different RL algorithms, which is currently limited by the exponential growth of the state space. Transfer learning strategies and better use of space invariance and symmetries are potential solutions.

Additionally, the lack of appropriate objective function, designing models with high generalization ability and the complexity and resource requirements of large-scale sequence alignments are the challenges that need to be addressed in future research. To conclude, RL based approaches have the potential to significantly enhance the reliability and effectiveness of MSA in bioinformatics.

References

1 M. Chatzou, C. Magis, J. M. Chang, C. Kamena, G. Bussotti, I. Erb, C. Notredame, ‘Multiple sequence alignment modeling: methods and applications’, *Briefings in Bioinformatics*, vol. 17, no. 6, pp. 1009–1023, Nov. 2016, doi: 10.1093/bib/bbv099.

2 R. V. Noorden, B. Maher, and R. Nuzzo, ‘Nature explores the most-cited research of all time.’ *The top 100 papers*. Nature 2014;514:550–3.

3 J. D. Thompson, D. G. Higgins, and T. J. Gibson, ‘CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice’, *Nucleic Acids Res* 1994;22:4673–90.

4 S. R. Eddy, ‘A Probabilistic Model of Local Sequence Alignment That Simplifies Statistical Significance Estimation’, *PLoS Computational Biology*, vol. 4, no. 5, p. e1000069, May 2008, doi: 10.1371/journal.pcbi.1000069.

5 C. Lee, C. Grasso, and M. F. Sharlow, ‘Multiple sequence alignment using partial order graphs’, *Bioinformatics*, vol. 18, no. 3, pp. 452–464, Mar. 2002, doi: 10.1093/bioinformatics/18.3.452.

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, Mar. 1970, doi: 10.1016/0022-2836(70)90057-4.

7 T. F. Smith and M. S. Waterman, ‘Identification of common molecular subsequences’, *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, Mar. 1981, doi: 10.1016/0022-2836(81)90087-5.

8 P. Hogeweg and B. Hesper, ‘The alignment of sets of sequences and the construction of phyletic trees: An integrated method’, *Journal of Molecular Evolution* vol. 20, no. 2, pp. 175–186, Jun. 1984, doi: 10.1007/BF02257378.

9 C. Notredame, D. G. Higgins, and J. Heringa, ‘T-coffee: a novel method for fast and accurate multiple sequence alignment 1 1Edited by J. Thornton’, *Journal of Molecular Biology*, vol. 302, no. 1, pp. 205–217, Sep. 2000, doi: 10.1006/jmbi.2000.4042.

10 C. B. Do, M. S. P. Mahabhashyam, M. Brudno, and S. Batzoglou, ‘ProbCons: Probabilistic consistency-based multiple sequence alignment’, *Genome Research*, vol. 15, no. 2, pp. 330–340, Feb. 2005, doi: 10.1101/gr.2821705.

11 I. M. Wallace, O. Orla, and D. G. Higgins, ‘Evaluation of iterative alignment algorithms for multiple alignment’, *Bioinformatics*, vol. 21, no. 8, pp. 1408–1414, Apr. 2005, doi: 10.1093/bioinformatics/bti159.

12 K. Katoh, ‘MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform’, *Nucleic Acids Research*, vol. 30, no. 14, pp. 3059–3066, Jul. 2002, doi: 10.1093/nar/gkf436.

13 R. C. Edgar, ‘MUSCLE: a multiple sequence alignment method with reduced time and space complexity’, *BMC Bioinformatics*, vol. 5, no. 1, p. 113, 2004, doi: 10.1186/1471-2105-5-113.

14 F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Soding, J. D. Thompson and D. G. Higgins, ‘Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega’, *Molecular Systems Biology*, vol. 7, no. 1, p. 539, Jan. 2011, doi: 10.1038/msb.2011.75.

15 L. Wang and T. Jiang, ‘On the Complexity of Multiple Sequence Alignment’, *Journal of Computational Biology*, vol. 1, no. 4, pp. 337–348, Jan. 1994, doi: 10.1089/cmb.1994.1.337.

16 C. Kemena and C. Notredame, ‘Upcoming challenges for multiple sequence alignment methods in the high-throughput era’, *Bioinformatics*, vol. 25, no. 19, pp. 2455–2465, Oct. 2009, doi: 10.1093/bioinformatics/btp452.

17 D. J. Lipman, S. F. Altschul, and J. D. Kececioglu, ‘A tool for multiple sequence alignment.’, *Proceedings of the National Academy of Sciences. U.S.A.*, vol. 86, no. 12, pp. 4412–4415, Jun. 1989, doi: 10.1073/pnas.86.12.4412.

18 J. D. Thompson, P. Koehl, R. Ripp, and O. Poch, ‘BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark’, *Proteins*, vol. 61, no. 1, pp. 127–136, Jul. 2005, doi: 10.1002/prot.20527.

19 R. S. Sutton and A. G. Barto, Reinforcement learning: an introduction, Second edition. in *Adaptive computation and machine learning series*. Cambridge, Massachusetts: The MIT Press, 2018.

20 J. Clifton and E. Laber, ‘Q-Learning: Theory and Applications’, *Annual Review of Statistics and Its Application*, vol. 7, no. 1, pp. 279–301, Mar. 2020, doi: 10.1146/annurev-statistics-031219-041220.

21 R. J. Williams, ‘Simple statistical gradient-following algorithms for connectionist reinforcement learning’, *Machine Learning* 8(23)

22 V. Mnih , A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P Lillicrap, D. Silver, K. Kavukcuoglu., ‘Asynchronous Methods for Deep Reinforcement

Learning', *International Conference on Machine Learning* 2016.
23 F.-M. Luo, S. Jiang, Y. Yu, Z. Zhang, and Y.-F. Zhang, 'Adapt to Environment Sudden Changes by Learning a Context Sensitive Policy', *The Association for the Advancement of Artificial Intelligence*, vol. 36, no. 7, pp. 7637–7646, Jun. 2022, doi: 10.1609/ aaai. v36i7.20730.
24 R. Bellman, *Dynamic programming*. Princeton, NJ: Princeton Univ. Pr, 1984.
25 V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller., 'Playing Atari with Deep Reinforcement Learning', *arXiv preprint* arXiv:1312.5602, 2013.
26 J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, T. Rocktäschel., 'A Survey of Reinforcement Learning Informed by Natural Language'. *arXiv*, Jun. 10, 2019. Accessed: Jul. 25, 2023. Available: http://arxiv.org/abs/1906.03926
27 I.-G. Mircea, I. Bocicor, and G. Czibula, 'A Reinforcement Learning Based Approach to Multiple Sequence Alignment', in *Soft Computing Applications*, vol. 634, V. E. Balas, L. C. Jain, and M. M. Balas, Eds., in Advances in Intelligent Systems and Computing, vol. 634. , Cham: Springer International Publishing, 2018, pp. 54–70. doi: 10.1007/978-3-319-62524-9_6.
28 R. Kinattinkara Ramakrishnan, J. Singh, and M. Blanchette, 'RLALIGN: A Reinforcement Learning Approach for Multiple Sequence Alignment', in *2018 IEEE 18th International Conference on Bioinformatics and Bioengineering (BIBE)*, Taichung, Taiwan: IEEE, Oct. 2018, pp. 61–66. doi: 10.1109/BIBE.2018.00019.
29 R. Jafari, M. M. Javidi, and M. Kuchaki Rafsanjani, 'Using deep reinforcement learning approach for solving the multiple sequence alignment problem', *SN Applied Sciences* vol. 1, no. 6, p. 592, Jun. 2019, doi: 10.1007/s42452-019-0611-4.
30 Y. Zhang, Q. Zhang, Y. Liu, M. Lin, and C. Ding, 'Multiple Sequence Alignment based on deep Q network with negative feedback policy', *Computational Biology and Chemistry*, vol. 101, p. 107780, Dec. 2022, doi: 10.1016/j.compbiolchem.2022.107780.
31 X. Xiang, D. Zhang, J. Qin, and Y. Fu, 'Ant Colony with Genetic Algorithm Based on Planar Graph for Multiple Sequence Alignment', *Information Technology J.*, vol. 9, no. 2, pp. 274–281, Feb. 2010, doi: 10.3923/itj.2010.274.281.
32 H. Carroll*, W. Beckstead, T. O'Connor, M. Ebbert, M. Clement, Q. Snell, et D. McClellan.*, 'DNA reference alignment benchmarks based on tertiary structure of encoded proteins', *Bioinformatics*, vol. 23, no. 19, pp. 2648–2649, Oct. 2007, doi: 10.1093/bioinformatics/btm389.
33 C. Kanz, 'The EMBL Nucleotide Sequence Database', *Nucleic Acids Research*, vol. 33, no. Database issue, pp. D29–D33, Dec. 2004, doi: 10.1093/nar/gki098.