Multiple Sequence Alignment using Deep Reinforcement Learning

Roman Joeres[1]

Abstract: Multiple sequence alignment (MSA) is one of the primal problems in biology and bioinformatics. The question of how to align multiple sequences correctly is crucial for many other fields of research, e.g., gaining information about the evolutionary distance of two or more sequences and therefore about their corresponding species, finding protein targets for drugs, or finding a drug for a certain target protein.

Reinforcement learning (RL), and especially deep reinforcement learning (DRL), has become popular in recent years. To name just a few, DRL has shown major success in complex games such as Atari Games, Chess, and Go.

We model the problem of aligning multiple sequences as a Markov decision process (MDP) and examine the performance of different (D)RL algorithms compared to state-of-the-art tools.

Keywords: Bioinformatics; Multiple Sequence Alignment; Reinforcement Learning; Deep Reinforcement Learning

1 Introduction

MSA is one of the most important open fields in bioinformatics because it has applications in nearly every other field of bioinformatics research. Some of the most intuitive examples are phylogenetic trees that are used to describe the evolutionary distance between species. This can be measured relative to the number of similar or dissimilar patterns between sequences of the compared species [FD87]. Other applications are drug design and drug development where similar regions of sequences are matched and information on secondary structures and functionalities are inferred [Ko06; Li10].

Because this problem is NP-complete, the optimal alignment cannot be computed directly [WJ94]. Most classical approaches use a biological fine-tuned scoring function to score alignments computed using dynamic programming or fast Fourier transformation to reproduce the real evolutionary changes between sequences [Ed04; Ka02; THG94]. Aside from these, some new ideas use techniques from machine learning to find good or optimal alignments. RL provides techniques to solve multiplayer games or problems of sequential decision-making by learning and interacting in an MDP. The key idea in such settings is

[1] Saarland University, Saarland Informatics Campus (GradSchool), Campus E1 3, 66123 Saarbrücken, Germany s8rojoer@stud.uni-saarland.de

to have an agent learning by trial-and-error in the MDP [SB18]. RL has been successfully applied to the traveling salesman problem [Ag19], solving the Rubik's Cube [KVW18], or optimizing a car manufacturing process [GGW20]. With methods from RL, DeepMind's AlphaFold won the CASP13 competition [Mo95], a comparison of different protein folding prediction programs that tries to find the correct fold of a protein structure [Se20].

In this paper, we give a comprehensive analysis of the performance of different RL algorithms in MSAs and offer a variety of interesting and promising ideas for future improvements. We use an MDP proposed by Mircea et al. [MBD14] who already received interesting results. At first, we will test multiple algorithms and optimization settings on DNA data from Mircea et al. Second, we test fewer algorithms on protein data from the BAliBASE dataset [TPP99].

2 Related Work

There are already approaches that apply RL to the problem of aligning multiple sequences. In 2014, Mircea et al. introduced RL to the field of MSAs [MBD14]. They focused on a tabular approach and proposed a first MDP to tackle the problem. Their work had promising results on some test data, and in a second paper with a slightly modified scoring function, the results were improved a bit [MBC16]. Jafari et al. introduced DQNs and an actor-critic algorithm in their work and could further improve the results of Mircea et al. in [JJR19]. But both papers suffer from the problem of a small selection of data to test their performance. In addition, both did not publish their code to reproduce the results and test it on larger benchmarks like BAliBASE [TPP99].

Other researchers also investigated the possibility to use RL techniques for MSAs using other models to represent sequence alignments. Edelkamp et al. [ET15] used a modified version of Monte Carlo tree search (MCTS) and achieved good results on some alignments from BAliBASE. Ramakrishnan applied DRL to the model of the Phylo game [Ka12]. But this model is too detailed to be applied to real-world problems with dozens of sequences and hundreds of molecules.

3 Multiple Sequence Alignment

The goal in MSA is to find mutations between sequences that model the evolutionary process the best. To test how well algorithms model this evolutionary process, benchmark datasets collect "true", hand-crafted solutions of alignments of multiple sequences. Before defining the different scorings in this field, we first define sequences and alignments formally.

Definition 3.1 (Sequence) *A sequence seq is a string over an alphabet Σ representing molecules M. For protein sequences, this alphabet contains 20 letters for the 20 amino acids (the molecules). For DNA sequences holds $\Sigma_{DNA} = \{A, C, G, T\}$, representing the four nucleic bases.*

Definition 3.2 (Alignment) *We denote an aligned sequence as seq', a string over $\Sigma' = \Sigma \cup \{-\}$. The gaps, $-$, represent mutations such as insertions and deletions between sequences. Let SE be the set of all sequences of one alignment problem instance and SE' be the set of all seq'. Then the alignment $A \subsetneq SE'$ contains possible aligned sequences and must fulfill three properties: (i) $\forall\ seq_1, seq_2 \in A : |seq_1| = |seq_2|$, (ii) $\forall\ seq' \in A : seq' \setminus \{-\} = seq$, and (iii) $\nexists\, i \leq |A| : \forall\ seq' \in A : seq'_i = -$.*

To align two or more sequences and to interpret the results of an alignment process, we have to score the resulting alignments to evaluate the quality of an alignment. Therefore, different scores have been developed and each of them has advantages and disadvantages as we discuss in the following.

Optimization Scores The sum-of-pairs-score (SP-score) is the most used score to align sequences. Used in a dynamic programming approach, one can compute the optimal alignment of k sequence of length at most n in $O(n^k)$ [NW70]. The SP-score for alignments $SP : SE' \longrightarrow \mathbb{R}$ is defined using the function $sp : M \times M \longrightarrow \mathbb{R}$ to score two molecules.

$$SP(A) = \sum_{i=1}^{N} \sum_{j=1}^{n-1} \sum_{k=i+1}^{n} sp\left(c_i^j, c_i^k\right) \quad \text{with} \quad sp\left(c_i^j, c_i^k\right) = \begin{cases} 0 & \text{if } c_i^j = c_i^k = - \\ 2 & \text{if } c_i^j = c_i^k \\ -1 & \text{if } c_i^j = - \text{ or } c_i^k = - \\ -2 & \text{if } c_i^j \neq c_i^k \end{cases}$$

measures the quality of aligned pairs of bases by assigning weights to types of pairs. We follow the definition of Mircea et al. [MBD14]. This kind of score is also used in state-of-the-art tools, namely CLUSTAL [THG94], MAFFT [Ka02], and MUSCLE [Ed04], which we later compare ourselfes to. Besides the SP-score, we use a relative score, called *column score* (C-score). This score $C : SE' \longrightarrow [0, 1]$ is defined as

$$C(A) = \frac{\text{\# perfectly aligned columns in A}}{\text{\# columns in A}}.$$

Evaluation Scores Databases like the BAliBASE offer hand-crafted and hand-optimized solutions for alignment problems and from the supervised learning point of view, those solutions are the label for the problem instance. For the evaluation of an algorithm, one can compare the output of the algorithm with the labeled alignment $R \subsetneq SE'$ using the Q-score $Q : SE' \times SE' \longrightarrow [0, 1]$ and TC-score $TC : SE' \times SE' \longrightarrow [0, 1]$ [Ed04].

$$Q(A, R) = \frac{\text{\#correctly aligned pairs in A}}{\text{\#aligned pairs in } R}$$

$$TC(A, R) = \frac{\text{\#correctly aligned columns in A}}{\text{\#columns in } R}$$

3.1 Markov Decision Process

An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{S}_0)$. The state-space $\mathcal{S}$ consists of all states and the action-space $\mathcal{A}$ represents all actions. The transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ describes to which states on can get. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ maps states and actions to rewards. $\mathcal{S}_0$ is the initial state of the problem.

For us, $\mathcal{S}$ defines the possible orders of the sequences, including incomplete orders when not all sequences are aligned. These sequences are subsets of SE'. The applicable actions $\mathcal{A}$ remain in every state the same, i.e., every sequence can be aligned to the current alignment. The transition function $\mathcal{T}$ is, therefore, deterministic and maps a state and an action to the state representing the resulting order of sequences when aligning sequences (performing the actions) from the start state. Using this definition, a state can also be defined as the order of sequences (sequence of actions) that lead to a state. The rewards $\mathcal{R}$ are defined using a scoring function $score : \mathcal{S} \rightarrow \mathbb{R}$ used for optimization.

$$R(s, a_i, s') = \begin{cases} 0 & \text{if } n = 1 \\ -\infty & \text{if } a_i \in s \\ score(s') & \text{otherwise} \end{cases}$$

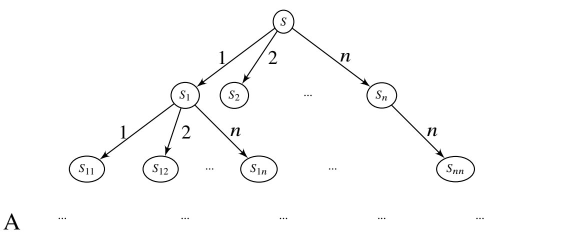
where $a_i \in s$ means that the i-th action was already performed earlier, i.e., the sequence i has already been aligned in s. The initial state $\mathcal{S}_0$ is the empty order, i.e., when no sequence is aligned. This model is visualized in Figure 1A.

4 Reinforcement Learning

RL is, besides supervised and unsupervised learning, the third big paradigm in machine learning. In contrast to supervised learning, RL does not depend on the existence of datasets. This learning technique is a trial-and-error approach that learns from the returns received from the environment, as depicted in Figure 1B. For each action the agent performs in the environment, it observes a reward that is used in reward functions to determine the quality of the selected action. The overall goal is to find a good policy, i.e., rules on how to behave in a state of the environment.

As in the papers of Mircea et al. [MBC16] and Jafari et al. [JJR19] we implement tabular agents (using the SARSA algorithm) as well as deep q-networks (DQNs) and apply both with three different reward structures, namely MC return, TD return, and λ-return. Jafari et al. used also an actor-critic approach with LSTM-cells. We will use the advantage actor-critic (A2C) consisting of an actor using policy gradients and a critic approximating the state-value function. To see the performance of policy gradients alone, we added the REINFORCE algorithm with and without a baseline to the portfolio of algorithms that are tested on MSAs.

We also implement an algorithm called "Upper confidence bounds applied to trees" (UCT) which is an improvement of normal MCTS. This algorithm also performs random rollouts



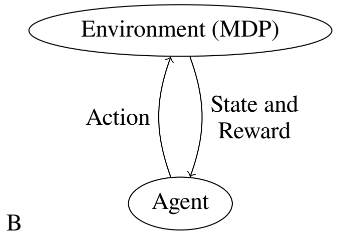


Fig. 1: A: Schematic overview of the model we use to describe multiple sequence alignments. B: Schematic view of a Markov decision process with the interactions between a learning agent and the environment.

in the state space to find a good action but improves its search using a tree policy to select a node in the known state space. The tree policy is given by

$$\underset{a}{max}\left\{v_a + \sqrt{\frac{\ln n}{n_a}}\right\} \tag{1}$$

where v_a is the average reward of action a, n is the total number of rollouts in the whole tree, and n_a the number of rollouts through this action. This formula is used to decide which child node to explore next by maximizing the score of the actions. If the child has not been explored yet, it expands the search tree in this node and performs a random rollout in one of the children [Co06].

5 Evaluation and Results

In the first step, we focus on the data that were already used by Mircea et al. [MBC16]. In Figure 2A, we can see exemplary results of the agents on the OxBench 429 data. This alignment contains 12 DNA sequences of length around 170 bases. For this task, our agents optimized the SP score and could outperform the classical reference tools as well as the results from Mircea et al. and Jafari et al.

One problem of SP-score optimization is the drop in performance in the alignments of the UCT-agent. The second part of the tree-policy (Equation 1) is most likely around 1, while the first part only depends on the reward (SP-score) of the alignments. Because the SP-score is unbounded, the first part dominates the formula. Here it can happen that the tree-policy always selects the same child because the SP-score of the first random rollout in the other children was too low. To solve this problem, we scaled the SP-score into the interval from 0 to 1. This is done using the center-star algorithm, which is proven to be a 2-approximation [Gu93]. The result is then used to estimate the upper bound, the lower bound is estimated as the highest score of 10 randomly arranged alignments. Using this trick, we could improve the SP-scores of UCT-alignments to a competitive level (see Table 1).

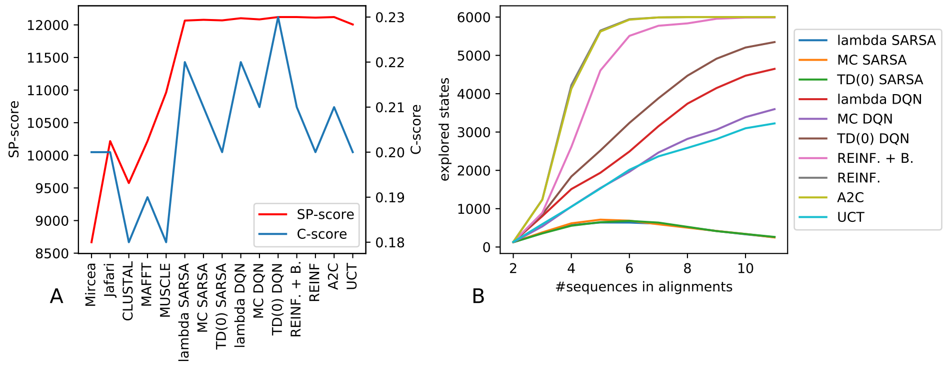


Fig. 2: Both plots show results from the OxBench 429 dataset. On the left, a comparison of the SP-scores and the C-scores of the 10 different agents and the reference algorithms. On the right, the number of explored states in the MDP is shown. The upper bound is given by the number of epochs (6000). The reference algorithms are not shown, because the numbers cannot be measured.

UCT-Mode	Hep.-C	P. Anubis	Ox469	Ox429	LGM	RLO	Ds. 1
Normal	18627	18848	639	11553	342	488	167
Adjusted	18627	18875	672	12006	348	488	167
Improvement	0.0%	0.1%	5.2%	4.0%	1.8%	0.0%	0.0%

Tab. 1: Improvement of adjusted UCT search with scaled rewards over normal UCT search with no scaling of the rewards.

	CLUSTAL	MAFFT	MUSCLE	SARSA	DQN	REINF.	A2C	UCT
Ox433	268	290	24	**353**	**353**	**353**	**353**	**353**
Ox641t2	659	723	442	**1053**	**1053**	**1053**	**1053**	**1053**
Ox34	355	377	-206	1215	**1223**	**1223**	**1223**	1208

Tab. 2: First protein analysis with data from the OxBench dataset.

In Figure 2B, we can see the number of unique explored states during the training. In contrast to tools like CLUSTAL, MAFFT, and MUSCLE, the presented agents perform many more alignments. From this, it follows that also the runtimes are much longer than for the classical tools. While CLUSTAL needs 0.06 seconds to find an alignment in OxBench 429, the SARSA agents need 20 minutes, and the policy agents up to 7 hours.

To extend this limited amount of data and to get an insight on how the tools might perform on protein sequence alignments, we choose three protein alignments from the OxBench dataset. OxBench focuses mainly on alignments between sequence families [Ra03]. The alignments consist of 3 and 6 sequences with lower sequence similarity but similar length compared to the DNA sequences. For those additional alignments, the results can be seen in table 2.

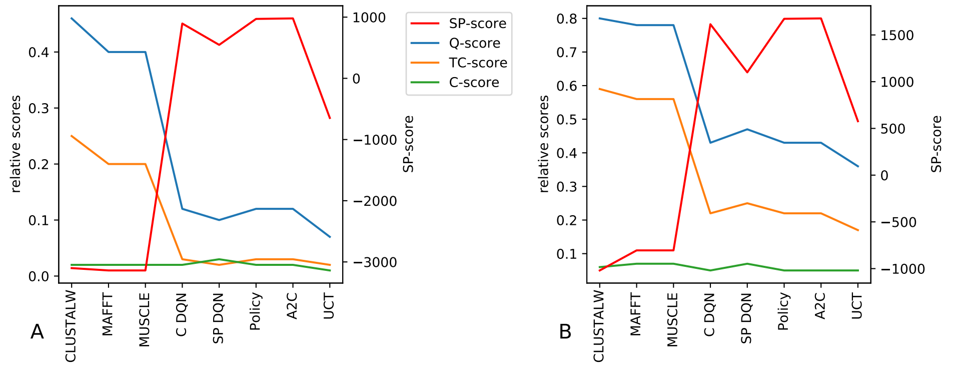


Fig. 3: Comparison of the performances of the algorithms with respect to different scoring functions. The values are averaged over the used alignments from the reference set 1.1 (left) and reference set 1.2 (right) of the BAliBASE dataset [TPP99].

In the second part of the analysis, we use alignments from the BAliBASE dataset [TPP99]. BAliBASE is made of reference subsets that contain sequences with similar properties. For each alignment in BAliBASE, there is a labeled alignment of the sequences. For our analysis, we use from the first reference set the alignments with fewer than or exactly ten sequences and an average length of the sequences of at most 500. These bounds are chosen because with longer sequences or more sequences the execution of RL algorithms takes too long. Those sequences can be split into two groups, namely the reference sets 1.1 and 1.2 containing alignments with lower and higher sequence similarities, respectively.

The comparison in this field is a bit different from the one before. Instead of analyzing an exemplary single sequence file, we average over the results within a reference subset. As the computation of these alignments takes much longer than aligning the few sequences above, we focus on DQN using the λ-return, the REINFORCE algorithm, the A2C algorithm, and the UCT agent. All of those agents will only optimize for the SP-score, except the DQN, it also optimizes for the C-score. We choose those agents because of their performance on the DNA data.

In Figure 3 the comparison of the average scores is denoted. Here, we additionally see the Q-scores and TC-scores of the alignments. The classical tools outperform the RL agents in terms of the Q-score and the TC-score due to the more fine-tuned scoring. On the other hand, the RL algorithms receive better SP-scores as they explicitly optimize for them. Again, this metric is evaluated using our simpler definition. We can also see that the RL tools narrow the gap to the state-of-the-art tools when the sequences to align are more similar as this is the case in the reference set 1.2 (Figure 3B).

6 Discussion

In Figure 2A, there is no big difference within the RL agents we implemented, because all of them work very similarly and the alignments are not complex. There are only a few DNA sequences, those are very similar and vary not much in length. The difference between our agents and the other RL agents is caused by the optimization score of the alignments. The agents of Mircea et al. and Jafari et al. optimized the C-score, not the SP-score, but the C-score is not a good measure for optimization. It automatically shrinks towards zero when there are more sequences or more different molecules in the type of sequence. The blue lines in Figure 2 show the optimization of the C-score and emphasize this problem. There are no big differences in the C-scores between the different agents and the scores are very low.

The results from the classical tools are optimized for their own SP-score, which is highly different from the simple one we use as described in section 3. In CLUSTALW, among others, this is done by sequence weighting based on the (dis-)similarity of a sequence to other sequences. The higher the dissimilarity or divergence of a sequence compared to others, the higher the sequence weight. These weights are used as multiplication factors for the sequences. There are several other improvements to this plain SP-scoring function presented above [THG94]. For the comparison, we computed our SP-scores of the alignments; therefore, those results are not very expressive. To get a more powerful result, one has to evaluate the alignments also using the SP-score function of the tools or know the labeled alignments as we do for the protein data.

When it comes to runtime, we see that the RL agents take much more time than the state-of-the-art tools caused by the increased number of alignments they have to compute. The reason for those additional alignments is rooted in the idea of RL and its trial-and-error character. To learn, an agent has to train for many episodes in its environment. Transferred to multiple sequence alignment, the agent has to perform many alignments to find out which are good ones and which are not that good.

The first results on the OxBench protein alignments are promising. Nevertheless, they are not very expressive, because the properties of the alignments are very similar to the ones from the dataset of Mircea et al. The pairwise sequence similarity is higher than in the BAliBASE dataset, the number of sequences is low, and the evaluation is done without labeled alignments. So, it is not very surprising that the RL tools perform better than the state-of-the-art tools. Furthermore, we can see that the adjusted UCT-algorithm still has some shortcomings. We see the scaling improves the performance (Table 1), but this does not always work (Table 2), as the estimates of the bounds might not be accurate enough.

The low performance has several reasons. If we inspect the alignments and in which order the sequences are aligned, we can get some additional insights in the alignment quality and how they were computed. The fine-tuned SP-score makes some implicit assumptions, such as one should align similar sequences first. From those assumptions, the alignments get a structure, i.e., the order of how the sequences are aligned. They produce complex alignment-trees with the sequences in its leaves and the alignments of the corresponding leaves in the inner nodes. These structures are similar within the classical tools but cannot be found in the simpler, path-like alignments from the reinforcement agents. This might cause lower performance when it comes to the Q-score and TC-score.

An additional problem that is related to the way we defined the SP-score is the fragmentation of the sequences, i.e., there are many small gaps and many single nucleotides aligned in a longer segment of another sequence. This problem comes from the way we handle the gaps. In our definition, every gap gets the same penalty. Alternatively, extending an already existing gap is much more biologically meaningful and should therefore be punished less than opening a new gap as in convex gap costs [MBD14].

7 Conclusion

In this work, we applied multiple algorithms from the field of RL to the biological problem of multiple sequence alignments. We performed an extensive comparison with three state-of-the-art tools, namely CLUSTAL, MUSCLE, and MAFFT, and with other RL algorithms that were applied to multiple sequence alignments.

We tested ten different algorithms on several DNA sequence alignment problems and compared the results to other tools. We could easily reproduce results from other approaches that also used RL. Second, we applied the methods to alignments of protein sequences from the BAliBASE dataset.

We saw that RL performs well on sequences that have almost the same length. Unfortunately, for the DNA instances we used, no labeled solutions are available. So, we cannot compute Q-scores and TC-scores to see how well RL performs in terms of biological correctness. For sequences from the BAliBASE whose lengths are more different, the alignment quality sinks as multiple small gaps are interrupting the sequences in the final alignments. On most of the alignments, all state-of-the-art tools outperformed our agents.

But the major drawback of RL in multiple sequence alignments is the time that is needed to compute the alignments. Classical approaches solve the BAliBASE alignments within seconds or milliseconds; whereas the computations of our agents last for hours on the same sequence data. The reason for this is primarily based on the number of alignments a reinforcement agent computes, which is much higher than the number of alignments a classical alignment algorithm performs. For example, CLUSTAL performs mostly one progressive alignment that is then improved by a few iterative refinement steps, no matter how many sequences there are to align.

Although the results are not good, RL can help improve MSAs. Progressive alignments can be modeled as MDPs and RL can help distinguish good from bad alignment decisions. These new approaches should then be evaluated on expressive benchmarks such as BAliBASE with labeled alignments.

8 Future Work

To receive better Q-scores and TC-scores, one can use other variants of the SP-score for optimization. We use a SP-score with linear gap costs that lead to many single and small gaps in the alignment and multiple spread nucleotide islands. To prevent this, affine gap costs or the SP-Score implementations from state-of-the-art tools can be used.

One way to reduce the runtime and to increase the flexibility is to let the network generalize between alignments using additional data from the sequences and to modify the approach such that align-trees can be output. An idea could be to combine graphs and RL to directly compute align-trees. Therefore, each sequence is represented as a node in a fully connected graph. Then a graph neural network (GNN) can be used to choose an edge to contract by aligning the two nodes of the contracted edge. The resulting alignment structure can be more complex because alignment-trees can be produced. RL would be used to learn the optimal strategy that contracts edges.

9 Acknowledgement

This paper is based on my bachelor's thesis at the chair for modeling and simulation of Prof. Verena Wolf under the supervision of Timo P. Gros. I want to thank them for the opportunity to write my thesis on this topic and for their help during the development of the work.

Thanks also to Prof. Tobias Marschall at Heinrich-Heine University for being the co-supervising professor. His biological perspective and advice helped to interpret the results.

Bibliography

[Ag19] Agostinelli, F.; McAleer, S.; Shmakov, A.; Baldi, P.: Solving the Rubik's cube with deep reinforcement learning and search. Nature Machine Intelligence 1/8, S. 356–363, 2019.

[Co06] Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo tree search. In: International conference on computers and games. Springer, S. 72–83, 2006.

[Ed04] Edgar, R. C.: MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic acids research 32/5, S. 1792–1797, 2004.

[ET15] Edelkamp, S.; Tang, Z.: Monte-carlo tree search for the multiple sequence alignment problem. In: Eighth Annual Symposium on Combinatorial Search. Citeseer, 2015.

[FD87] Feng, D.-F.; Doolittle, R. F.: Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. Journal of molecular evolution 25/4, S. 351–360, 1987.

[GGW20] Gros, T. P.; Groß, J.; Wolf, V.: Real-Time Decision Making for a Car Manufacturing Process Using Deep Reinforcement Learning. In: 2020 Winter Simulation Conference (WSC). IEEE, S. 3032–3044, 2020.

[Gu93] Gusfield, D.: Efficient methods for multiple sequence alignment with guaranteed error bounds. Bulletin of mathematical biology 55/1, S. 141–154, 1993.

[JJR19] Jafari, R.; Javidi, M. M.; Rafsanjani, M. K.: Using deep reinforcement learning approach for solving the multiple sequence alignment problem. SN Applied Sciences 1/6, S. 592, 2019.

[Ka02] Katoh, K.; Misawa, K.; Kuma, K.-i.; Miyata, T.: MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic acids research 30/14, S. 3059–3066, 2002.

[Ka12] Kawrykow, A.; Roumanis, G.; Kam, A.; Kwak, D.; Leung, C.; Wu, C.; Zarour, E.; Sarmenta, L.; Blanchette, M.; Waldispühl, J. et al.: Phylo: a citizen science approach for improving multiple sequence alignment. PloS one 7/3, e31362, 2012.

[Ko06] Konagurthu, A. S.; Whisstock, J. C.; Stuckey, P. J.; Lesk, A. M.: MUSTANG: a multiple structural alignment algorithm. Proteins: Structure, Function, and Bioinformatics 64/3, S. 559–574, 2006.

[KVW18] Kool, W.; Van Hoof, H.; Welling, M.: Attention, learn to solve routing problems! arXiv preprint arXiv:1803.08475/, 2018.

[Li10] Li, Q.; Cheng, T.; Wang, Y.; Bryant, S. H.: PubChem as a public resource for drug discovery. Drug discovery today 15/23-24, S. 1052–1057, 2010.

[MBC16] Mircea, I.-G.; Bocicor, I.; Czibula, G.: A reinforcement learning based approach to multiple sequence alignment. In: International Workshop Soft Computing Applications. Springer, S. 54–70, 2016.

[MBD14] MIRCEA, I.-G.; BOCICOR, M.-I.; DÎINCU, A.: ON REINFORCEMENT LEARNING BASED MULTIPLE SEQUENCE ALIGNMENT. Studia Universitatis Babes-Bolyai, Informatica 59/2, 2014.

[Mo95] Moult, J.; Pedersen, J. T.; Judson, R.; Fidelis, K.: A large-scale experiment to assess protein structure prediction methods. Proteins: Structure, Function, and Bioinformatics 23/3, S. ii–iv, 1995.

[NW70] Needleman, S. B.; Wunsch, C. D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of molecular biology 48/3, S. 443–453, 1970.

[Ra03] Raghava, G.; Searle, S. M.; Audley, P. C.; Barber, J. D.; Barton, G. J.: OXBench: a benchmark for evaluation of protein multiple sequence alignment accuracy. BMC bioinformatics 4/1, S. 47, 2003.

[SB18] Sutton, R. S.; Barto, A. G.: Reinforcement Learning: An Introduction. The MIT Press, 2018.

[Se20] Senior, A. W.; Evans, R.; Jumper, J.; Kirkpatrick, J.; Sifre, L.; Green, T.; Qin, C.; Žídek, A.; Nelson, A. W.; Bridgland, A. et al.: Improved protein structure prediction using potentials from deep learning. Nature 577/7792, S. 706–710, 2020.

[THG94] Thompson, J. D.; Higgins, D. G.; Gibson, T. J.: CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic acids research 22/22, S. 4673–4680, 1994.

[TPP99] Thompson, J. D.; Plewniak, F.; Poch, O.: BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. Bioinformatics (Oxford, England) 15/1, S. 87–88, 1999.

[WJ94] Wang, L.; Jiang, T.: On the complexity of multiple sequence alignment. Journal of computational biology 1/4, S. 337–348, 1994.

Appendix

Tool	Setting
CLUSTAL	Version 1.2.4 from December 16th, 2016 (Clustal Omega)
MUSCLE	Version 7.271 from July 3rd, 2020 (fast and progressive *FFT-NS-2* option [Ka02] p.4)
MAFFT	Version 3.8.31

Tab. 3: All of the tools are executed in default mode, so no search-influencing parameters were set.

General	
Network	Input x 256 x 128x x 64 x output
Activations	Mix of ReLU and Softmax
epochs	50*# seqs

SARSA and DQN		Policy Gradients	
α	0.1	α	0.01
γ	0.9	γ	0.99
λ	0.1	Baseline/critic	state-value network
ϵ-start	1	UCT	
ϵ-end	0	simulations	50*# seqs
ϵ time	10%	rollouts	1
TD-steps	TD(0)	C	1

Tab. 4: Hyperparameters used in the training of the agents.

Everything was executes on an Ubuntu Server, Version 18.04.4 LTS (Bionic Beaver) with 8 kernels and 4 GHz clocking.
The code for this paper and the underlying thesis is available on GitHub (https://github.com/Old-Shatterhand/MSADRL).