FrameRL: DNA-Protein Sequence Alignment using Deep Reinforcement Learning

Wai Kei Li
College of Computer Studies
De La Salle University
Manila, Philippines
wai_kei_li@dlsu.edu.ph

Justin Ayuyao
College of Computer Studies
De La Salle University
Manila, Philippines
justin_ayuyao@dlsu.edu.ph

John Carlo Joyo
College of Computer Studies
De La Salle University
Manila, Philippines
john_carlo_joyo@dlsu.edu.ph

Renz Ezekiel Cruz
College of Computer Studies
De La Salle University
Manila, Philippines
renz_cruz@dlsu.edu.ph

Roger Luis Uy
College of Computer Studies
De La Salle University
Manila, Philippines
roger.uy@dlsu.edu.ph

ABSTRACT

Aligning DNA and protein sequences in three reading frames, referred to as Three-Frame Alignment, is a dynamic programming (DP) approach for the alignment between a reference protein sequence and an input DNA sequence. Despite finding optimal alignments with the usage of three matrices, the memory usage of Three-Frame Alignment scales with the lengths of the sequences, making longer reads costly. This paper proposes FrameRL, a deep reinforcement learning approach with an environment based on Zhang's Three-Frame alignment algorithm and agent that can perform DNA-Protein sequence alignment. The resulting approach showed better scalability in memory usage for longer reads, resulting in an overall space complexity of O(1) compared with O(MN) of dynamic programming approaches, with M and N as the lengths of the DNA and protein sequences, respectively.

KEYWORDS

Three-frame alignment, global alignment, deep reinforcement learning, deep Q-learning, agent, environment, dueling double deep Q-networks

1 INTRODUCTION

Sequence alignment is the process of aligning nucleotide or amino acid sequences to identify common regions in order to find commonalities in ancestry, structure, function, and others [1]. Directly translating the DNA and finding the optimal alignments among all other possible amino acid counterparts is computationally heavy; therefore, dynamic programming approaches like the Needleman-Wunsch [5] and Smith-Waterman [8] algorithms were often utilized. Despite finding an optimal alignment, dynamic programming approaches are limited by their space complexity, which is directly proportional to the product of the lengths of each of the sequences. Therefore, these approaches are preferred for shorter sequence reads, as handling and recording large matrices is computationally costly.

Works like the DQNalign [9] and EdgeAlign [3] have already implemented reinforcement learning into sequence alignment, specifically, DNA-to-DNA. DQNalign was the first among the two and proved the viability of using deep reinforcement learning for DNA-to-DNA sequence alignment. It involves converting the sequence alignment task into a sliding windows environment with one window per sequence. Each window has a size of N representing a subsequence of size N. Their agent will then perform alignment on these windows and move them until each of them reaches the end of their respective sequences.

Zhang's Three-Frame algorithm uses three reading frames and three matrices (I, D, and C), effectively detecting and adjusting for frameshifts. The dynamic programming approach utilizes several recurrence relations to find the best score possible from the three matrices. The I matrix looks into the max score given an insertion. The C matrix takes charge of looking for the score of the best scores overall, and the D matrix is for deletion.

Like DQNalign, this paper proposed a deep reinforcement learning approach to DNA and protein sequence alignment to solve the space complexity issues associated with Zhang's three-frame alignment. The proposed method leverages the benefits of reinforcement learning while still attempting to recreate the approach used in Zhang's three-frame alignment. The proposed method was able to distinguish perfect matches and several frameshift errors but still experiences instability for mismatches or substitutions. It was also found to have better memory scalability when more extensive sequences are involved.

Section 2 of this paper discusses the training procedure, the agent and environment, the network architecture, and some experimentations performed. Space complexity, query tests, and agent training progression are discussed in Section 3. Lastly, conclusions and future work are presented in Section 4.

2 DNA TO PROTEIN ALIGNMENT WITH REINFORCEMENT LEARNING

Zhang's Three-Frame approach [12] is a global alignment dynamic programming solution providing the optimal alignment between the DNA and protein sequences. However, the issue with the approach stems from the part where it handles three matrices and its constant traversal for the simple act of attempting to gather the score. The space complexity of such an ordeal is also in O(MN) space, with M being the length of the DNA and N being the length of the protein. This makes it difficult to scale when the inputs get into millions in length.

The traversal of the three matrices creates a potential pitfall regarding time and space complexity in long reads. The usage of the Deep Reinforcement Learning method, or DRL, aims to solve the problem of scalability due to the neural networks having a time complexity of O(N), where N is the length of the query sequence. The speedup will not be immediately realized due to the huge overhead that the neural network demands. However, it performs better for the space complexity aspect as the total space complexity of the neural network would be O(1) [3]. The overall space complexity would still be O(M+N) due to loading in the reference nucleotide and input proteome, with M as the DNA length and N as the protein sequence length. Still, it is not comparable to Zhang's Three-Frame, wherein the space complexity is still O(MN) multiplied by three due to the I, C, and D matrices.

2.1 Training Procedure

To successfully train an unbiased RL agent, diverse input data sets are necessary to avoid biases associated with limited or overly specific datasets.

The process was initiated by generating a DNA sequence of length 1000. This training length was chosen to expose the agent to as many nucleotides and codons as possible without hindering training time. The first sequence is generated by randomly selecting nucleotides (A, G, C, or T). Then, the mutations were introduced using the JC69 model [2] to the first sequence to create the second sequence. On top of that, insertions and deletions are introduced in both sequences, so the RL agent is exposed to a wide variety of sequences through the Zipfian distribution-based indel length model [7]. Finally, the second DNA sequence is translated into a protein sequence. This is repeated until five sets of DNA and protein pairs are generated.

In addition, random protein sequences were generated for each DNA sequence so that the agent is trained in scenarios wherein the sequences are highly mismatched and where insertions, deletions, and frameshifts occur more frequently. Five training sets were generated to train the agent, each containing one DNA sequence, one directly matched protein sequence, and one randomly generated protein sequence. For convenience and easier tallying and tracking of incorrect actions, the agent's reward system punishes the agent's reward score by -2 when it makes mistakes and rewards it with 0 for every correct action. This reward system has made the agent prioritize looking for and frameshift matches. However, it becomes unstable regarding in/del actions and mismatches. These instabilities also shift the reading frames, which primarily affects the accuracy of the alignment and influences succeeding actions.

2.2 Agent and Environment

The environment mainly consists of the reference proteome of size M and the target DNA sequence of size N, wherein the agent traverses both using respective pointers. The environment is considered a Sequential Partially Observable Markov's Decision Process [10], where it only allows the agent to see a part of the environment because the agent's action will affect the current reading frames. The environment can also be considered deterministic despite having multiple correct actions, and this is due to those actions being sub-optimal.

The main objective of the environment is to recreate the process of Three-Frame Alignment. The environment is deterministic, with many states due to the twenty-one proteins in the codon table and eight of them in a subset comprising the past and current three reading frames alongside their respective proteome. Effectively, 54 billion states (including the stop codon/protein) are possible. Thus, a brute-force approach for both methods is impractical and unfeasible.

Conversely, the agent is a separate entity that interacts with the environment using distinct actions: MATCH (M), FRAMESHIFT_1 (F1), FRAMESHIFT_3 (F3), INDEL, and MISMATCH. The actions F1, M, and F3 pertain to aligning the current protein with the current reading frames 1, 2, and 3, respectively. INDELs, on the other hand, are alignments between the previous protein and any of the current frames or current protein with any previous frames. Lastly, MISMATCH happens when all other actions are not possible; thus, only the substitution of frames is possible. The precedence of these actions is based on the understanding from the simplification of the recurrence relation of Zhang's Three-Frame alignment.

M actions are always preferred, followed by F1 and F3 actions, and then by INDELs before resorting to MISMATCH. This precedence came from calculating the scores in the recurrence relation wherein Matches are favored over Frameshift matches, which are then favored over Indel matches, and which are also favored over Mismatches or Substitution. Depending on the result, these actions move the DNA and protein pointers rightward. The agent is punished if the action it chooses is incorrect, but there are no rewards if the agent is correct.

For the learning stage, the agent employs epsilon-greedy exploration [10], wherein it does random actions when there are higher epsilon values to experience as many states, actions, and rewards as possible and possibly determine and find patterns. These actions eventually update the agent's network, and the epsilon value is decayed over time as the agent's network is further trained.

2.3 Network Architecture

The agent in this paper adopts a Dueling Double Deep Q-Network (DDDQN) architecture similar to the approach used in DQNalign [9] for determining its actions. By leveraging the DDDQN architecture, the agent can expedite its decision-making process and create better learning outcomes. This architecture is particularly beneficial in distinguishing states that have more significance in learning.

The choice of utilizing a convolutional neural network (CNN) in the agent's network architecture is influenced by previous researchers who employed it in their deep reinforcement learning tasks, specifically in DQNalign and EdgeAlign [3]. Notably, the inception of CNNs in DRL can be traced back to Google DeepMind's research [4], which used the Atari games as their environment.

In the subsequent layers of the network, the Dueling architecture was incorporated to effectively segregate the State-action and Action-advantage values, thereby facilitating a more accurate estimation of future Q values by the network.

2.4 Experimentation

2.4.1 Query Experiments. The capacity of the agent to perform alignment was tested on select proteomes from the organism Drosophila melanogaster [11] (Fruit Fly). For the experiment, ten proteomes were selected from the list of Fruit Fly proteomes with a given length of 333. After that, the selected proteins are directly translated into their respective DNA sequences. These DNA sequences will be aligned with several reference proteomes from the previously selected proteomes. A random protein is first selected to simulate a query, and a target substring of length M is also shattered from that protein. After that, the random protein's ID and the translated DNA are recorded. The protein ID will serve as a reference in identifying the parent or origin of the target substring. Afterward, the order of the ten random proteins is shuffled, and the translated DNA is aligned with each protein. By assumption, the protein with the highest alignment score should contain the target substring. This random selection of protein and target substring is tested for each of the ten selected proteins.

2.4.2 Memory Usage Experiments. A synthetic benchmark is performed by generating a DNA sequence of a desired length to serve as the reference. To generate the query, the reference is shattered with mutations incorporated into the sequence based on the JC69 model [2] before being converted to a protein sequence. The RL Agent and a sequential implementation in Python of Zhang's Three-Frame algorithm are then given these sequences to simulate alignment with varying read lengths. The benchmark is performed by aligning on base pair lengths of 10, 30, 60, 100, 300, 500, 800, 1000, 1500, 3000, 4500, 6000, 7500, 9000, 13500, and 15000. For each given base pair length, memory is sampled for every alignment step, which is then averaged to give a comparative memory usage given a base pair length. These results were then compiled, and a graph was generated to visualize the space complexity of the aligners.

3 RESULTS AND DISCUSSION

3.1 Agent Training

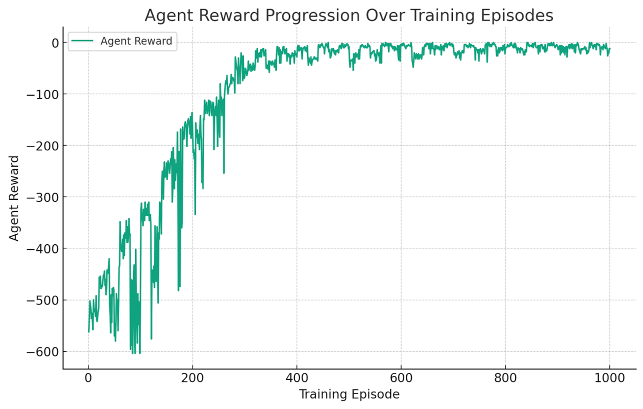

Figure 1: Agent Reward Progression

The training results in Figure 1 show that within the allotted 1000 episodes, the agent could learn the pattern of detecting perfect matches, frameshift_1 matches, and frameshift_3 matches. The first few hundred training episodes include substantial inaccuracy caused by the epsilon-greedy exploration training approach that makes random actions. Eventually, this epsilon value fully decays at episodes 400 to 500. Still, it retains a 1% chance of making a random action to facilitate the exploration even after epsilon has been fully decayed. The random actions prevent our agent from remaining in a local minima while training.

However, in relation to accuracy, even minor random errors in the agent's predictions will cause the reading frames to be shifted. So, in the long term, minor errors like these may cause a massive difference in the overall score but will not affect the result if the given sequences perfectly match.

```
Prev. Frames: _, _, _ => PP: *
Curr Frames: *, M, C => CP: M
Action: 0 (MATCH)
Reward of Action: 0

Prev. Frames: *, M, C => PP: M
Curr Frames: V, L, Y => CP: L
Action: 0 (MATCH)
Reward of Action: 0

Prev. Frames: V, L, Y => PP: L
Curr Frames: T, R, V => CP: R
Action: 0 (MATCH)
Reward of Action: 0
```

Figure 2: An example of the Alignment History

Since the primary approach attempts to mimic the original Three-Frame alignment and account for insertions and deletions, the agent's environment will be given six reading frames: three past frames and three present frames, as well as two proteins: one past protein (PP), and one current protein (CP) as displayed in Figure 2 and with the given information, the agent will predict the correct action to be done.

3.2 Memory Usage and Query Tests

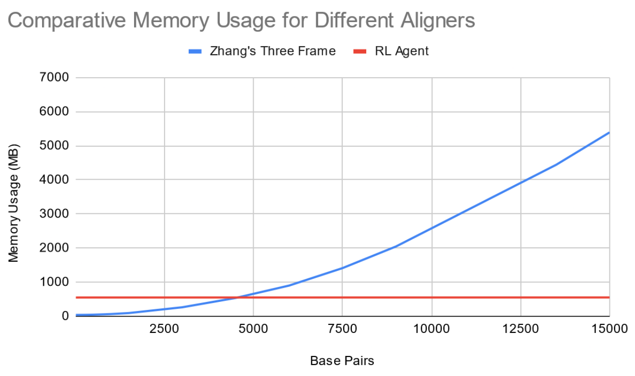

Figure 3: Memory Utilization of the Aligners

3.2.1 Memory Usage Comparison. Figure 3 presents a comparative analysis of the memory usage for the two different approaches in aligning: dynamic programming and reinforcement learning. The comparison was performed across various base pair lengths, emphasizing how each approach scales in terms of memory requirements.

As shown in Figure 3 and Table S2, the agent has a space complexity of O(1) since it only has to load the weights of the Q-network. Memory usage remained constant since the agent was able to take advantage of Python's buffered reader [6]. Zhang's three-frame aligner also takes advantage of the buffered reader, but despite that, the dynamic programming approach has to maintain three matrices, namely the I, D, and C matrix. This results in a space complexity of O(MN) where M is the length of the protein sequence and N is the length of the DNA sequence. For lengths beyond 150 base pairs, typical of long reads, the memory utilization of the dynamic programming approach quickly grows, making it unsuitable for alignment in long reads. Therefore, the agent outperformed the dynamic programming approach regarding memory usage since its space complexity remains constant for long reads. This makes the approach suitable for alignment tasks in devices with a constraint on memory footprint since the agent can perform alignment with a predictable memory utilization pattern.

3.2.2 Agent Query Tests. Table S1 displays the alignment scores for all query tests performed. The target protein represents the shattered sequence or substring of a source protein that the query will try to align with other proteins. The source proteins are also identified via their respective protein IDs. The respective columns represent alignment scores between the translated DNA sequence of the source protein and other proteins in the list. For example, in Row 1, the target protein is FVRIKQSLKP, originating from the protein Q7K1S. When aligning the translated DNA of Q7K1S with another protein whose ID is Q9V3Y7, the resulting alignment score is 79. From the table, it can be seen that the highest alignment scores are from the same protein IDs for each target protein. This further supports the assumption that the protein with the highest alignment score should contain the target substring. This indicates that the agent could fully identify matches between DNA and Protein sequences.

4 FUTURE WORKS

For future works, it is important to note that there are still minor instabilities when the agent encounters situations involving insertion, deletion, and substitution/mismatch. At the same time, further investigation and optimization are required for the incorrect truncation and lack of padding for the last reading frames in the environment. This is because the agent stops performing the alignment when no more current three reading frames are in the DNA sequence. Furthermore, the network architecture and layers could be further tested and optimized, for example, by changing the total number of layers, reconfiguring and replacing convolutional layers, etc.

The reinforcement learning model introduced in this paper is in its initial implementation. As such, it requires further refinement before it could be compared with alignment tools such as BLAST or algorithms like seed-chain-extend. Further refinements would include the optimization of the policy gradients and increasing the model's window size which will allow more reading frames to be evaluated during alignment.

In exploring potential RL models suitable for the deterministic environment of the paper, other models that offer performance enhancements can be considered. One promising model is the Deep Deterministic Gradient Policy (DDGP), which warrants further investigation due to its functionality in environments with continuous action spaces. Given the limitations of Q-learning in utilizing policy gradients effectively, alternatives that optimize through policy gradients could potentially yield a better agent. Additionally, the concept of imitation learning, where another model mimics and potentially outperforms the baseline model by learning from its actions in specific states, presents an interesting avenue for further exploration.

REFERENCES

[1] Stephen F Altschul and Mihai Pop. 2017. Sequence alignment. (2017).

[2] Thomas H Jukes, Charles R Cantor, et al. 1969. Evolution of protein molecules. *Mammalian protein metabolism* 3, 24 (1969), 21–132.

[3] Aryan Lall and Siddharth Tallur. 2023. Deep reinforcement learning-based pairwise DNA sequence alignment method compatible with embedded edge devices. *Scientific Reports* 13, 1 (2023), 2773.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[5] Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology* 48, 3 (1970), 443–453.

[6] Python. 2024. Built in Functions. https://docs.python.org/3/library/functions.html

[7] Bin Qian and Richard A Goldstein. 2001. Distribution of indel lengths. *Proteins: Structure, Function, and Bioinformatics* 45, 1 (2001), 102–104.

[8] Temple F Smith, Michael S Waterman, et al. 1981. Identification of common molecular subsequences. *Journal of molecular biology* 147, 1 (1981), 195–197.

[9] Yong-Joon Song, Dong Jin Ji, Hyein Seo, Gyu Bum Han, and Dong-Ho Cho. 2021. Pairwise heuristic sequence alignment algorithm based on deep reinforcement learning. *IEEE open journal of engineering in medicine and biology* 2 (2021), 36–43.

[10] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.

[11] UniProt. [n. d.]. Drosophila melanogaster (Fruit fly). https://www.uniprot.org/proteomes/UP000000803

[12] Zheng Zhang, William R Pearson, and Webb Miller. 1997. Aligning a DNA sequence with a protein sequence. In *Proceedings of the first annual international conference on Computational molecular biology.* 337–343.

5 SUPPLEMENTARY MATERIALS

Target Protein (Shattered)	Source Protein ID	ALIGNMENT SCORE (per protein ID)									
		Q7K1S1	Q9V3Y7	Q9VUJ0	Q9VMX3	Q9W0N1	Q9VXT2	Q9VN80	Q7JVH0	Q8IPF7	Q7KVY7
FVRIKQSLKP	Q7K1S1	1689	79	109	165	122	169	133	133	113	133
LPARLEDNSG	Q9V3Y7	115	1712	140	161	104	129	159	129	127	140
EQRRQRDAVG	Q9VUJ0	119	151	1724	152	104	135	117	125	127	143
LFAYRTEEST	Q9VMX3	170	99	107	1713	103	148	131	110	112	108
MHRLIFEVQQ	Q9W0N1	139	130	109	103	1712	118	152	135	113	147
RLLNLRRHQP	Q9VXT2	88	77	116	146	146	1761	122	141	87	128
NCRQLESLLL	Q9VN80	80	96	92	113	126	149	1725	97	110	99
EDEDFIKSVN	Q7JVH0	100	140	100	156	101	169	140	1681	161	173
ESDQQENSPD	Q8IPF7	105	91	138	125	114	136	121	121	1717	197
KGADELDQAE	Q7KVY7	123	144	102	169	118	137	141	167	143	1672

Supplementary Table 1: Alignment Score per Query Test

Target Protein (Shattered)	Source Protein ID	REWARD SCORE (per protein ID)									
		Q7K1S1	Q9V3Y7	Q9VUJ0	Q9VMX3	Q9W0N1	Q9VXT2	Q9VN80	Q7JVH0	Q8IPF7	Q7KVY7
FVRIKQSLKP	Q7K1S1	0	-26	-24	-34	-28	-22	-36	-18	-30	-32
LPARLEDNSG	Q9V3Y7	-18	0	-40	-22	-18	-28	-16	-14	-28	-24
EQRRQRDAVG	Q9VUJ0	-38	-28	0	-34	-32	-32	-34	-40	-22	-26
LFAYRTEEST	Q9VMX3	-40	-22	-34	0	-20	-18	-34	-32	-28	-36
MHRLIFEVQQ	Q9W0N1	-34	-30	-34	-36	0	-26	-32	-42	-26	-32
RLLNLRRHQP	Q9VXT2	-30	-20	-28	-26	-26	0	-26	-28	-30	-24
NCRQLESLLL	Q9VN80	-20	-26	-36	-20	-32	-28	0	-22	-28	-34
EDEDFIKSVN	Q7JVH0	-28	-26	-26	-24	-34	-24	-24	0	-32	-22
ESDQQENSPD	Q8IPF7	-38	-22	-22	-32	-20	-28	-32	-18	0	-32
KGADELDQAE	Q7KVY7	-30	-28	-18	-20	-28	-28	-16	-30	-30	0

Supplementary Table 2: Agent Reward Score per Query Test

Number of Base Pairs	Memory Usage (Bytes)		Memory Usage (Megabytes)	
	Zhang's Three Frame	RL Agent	Zhang's Three Frame	RL Agent
10	40169472	550031360	40.1695	550.03136
30	40169472	550031360	40.1695	550.03136
60	40427520	550526976	40.4275	550.526976
100	40545657.84	551022592	40.5457	551.022592
300	42661205.33	551025294.5	42.6612	551.0252945
500	46732925.57	551284736	46.7329	551.284736
800	57420523.65	551284736	57.4205	551.284736
1000	65449894.31	551284736	65.4499	551.284736
1500	95004232.9	551284736	95.0042	551.284736
3000	264655709.7	551412417.4	264.6557	551.4124174
4500	538028175.7	551555072	538.0282	551.555072
6000	900794987.2	551586211.6	900.7950	551.5862116
7500	1412761902	551825408	1412.7619	551.825408
9000	2046236231	551840228.1	2046.2362	551.8402281
13500	4441659236	552251106.6	4441.6592	552.2511066
15000	5391433074	552589310.8	5391.4331	552.5893108

Supplementary Table 3: Memory Usage Per Base Pair Counts