Highlights

DQNalign : A deep reinforcement learning algorithm for pairwise heuristic sequence alignment

Yong-Joon Song,Dong Jin Ji,Hyein Seo,Gyu-Bum Han,Dong-Ho Cho

- Define the agent and environment for applying the deep reinforcement learning algorithm to the sequence alignment system
- Investigate the effect of each parameter to the performance of sequence alignment
- Combine the conventional sequence alignment algorithm and deep reinforcement learning based extension algorithm to improve the coverage of global sequence alignment
- Prove how the proposed algorithm can achieve the identical results with optimal global alignment
- Suggest a novel approach to the sequence alignment method with broad scalabilty

DQNalign : A deep reinforcement learning algorithm for pairwise heuristic sequence alignment

Yong-Joon Song[a,1], Dong Jin Ji[a,2], Hyein Seo[a,3], Gyu-Bum Han[a,3] and Dong-Ho Cho[a,*]

[a]School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, South Korea

ARTICLE INFO

Keywords:
Sequence alignment
Pairwise alignment
Global alignment
Reinforcement learning
Deep Q-learning network
Model of evolution
Sequence comparison

ABSTRACT

Various methods have been developed to analyze the association between organisms and their genomic sequences. Among them, sequence alignment is most frequently used for comparative analysis of biological genomes. However, the traditional sequence alignment method is considerably complex in proportion to the length of the sequences, and it is considerably challenging to align lengthy sequences such as a human genome. Currently, several multiple sequence alignment algorithms are available that can reduce the complexity and improve the alignment performance of various genomes. However, there have been relatively fewer attempts to improve the alignment performance of the pairwise alignment algorithm. After grasping these problems, we intend to propose a new sequence alignment method using deep reinforcement learning. This research show how the deep reinforcement learning can be applied to the sequence alignment system and how the deep reinforcement learning can improve the conventional sequence alignment method.

1. Introduction

Recent advancements in sequencing technology have enabled the analysis of organisms with long sequences [1]. For organisms with short sequences, the evolutionary distances between the organisms could be easily analyzed using the older pairwise alignment methods, such as the conventional Needleman-Wunsch (NW) algorithm [2], and the relationship between the organisms could be investigated. However, since the conventional pairwise alignment method is extremely simple to determine the characteristics of genomic sequences, it is difficult to balance the complexity and performance. Therefore, various attempts have failed in long sequence alignment. The

*Corresponding author at : Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, 373-1, Guseong-dong, Yuseong-gu, Daejeon 305-701, South Korea. Fax: +82 42 869 4042. E-mail address: dhcho@kaist.ac.kr (D-H. Cho).

syjqkrtk@kaist.ac.kr (Y. Song); jdj0524@kaist.ac.kr (D.J. Ji); hiseo2@kaist.ac.kr (H. Seo); hgb1012@hanmail.net (G. Han); dhcho@kaist.ac.kr (D. Cho)

ORCID(s):

conventional dynamic programming-based NW alignment method has a complexity that is proportional to the product of the lengths of the two different nucleotide sequences, which results in great difficulty from the alignment perspective.

Thus far, the sequence alignment method has reduced the complexity through the development of a multiple sequence alignment (MSA) method that aligns multiple sequences simultaneously. This method has been improved through various approaches, such as detecting common subsequences, pairing methods, or increasing the speed by using multithreads based on GPUs [3-5]. However, there remains a critical point at which the complexity of pairwise alignment could be more severe in case of MSA [6].

In particular, several pairwise alignment algorithms, such as banded alignment, the BLAST, and the MUMmer have been proposed to improve the speed of pairwise alignment [7-9]. These alignment methods attempted to solve the complexity issue by limiting the range of the alignment and by extending the alignment after word matching or average common substring matching. However, in these cases, there were some accuracy problems in the process of extending short local alignments for large and complex sequences.

To improve the problems of conventional alignment methods, we proposed a novel alignment method using deep reinforcement learning agent. Before explaining our method, reinforcement learning is a way to teach an agent that could choose the best actions by observing an environment in a given system. The conventional tabular based reinforcement learning has a difficulty in expiring and learning the massive and complex systems. To improve this, the deep reinforcement learning method was proposed, and it overcomes the limitations by approximately learning the complex systems. The development of reinforcement leaning has shown amazing performance in various complex systems. So, we decided to apply this deep reinforcement learning method to sequence alignment system which is willing to find the optimal matches in two entire sequences. And rest of this paper, we will describe how can we apply the deep reinforcement learning in the sequence alignment system.

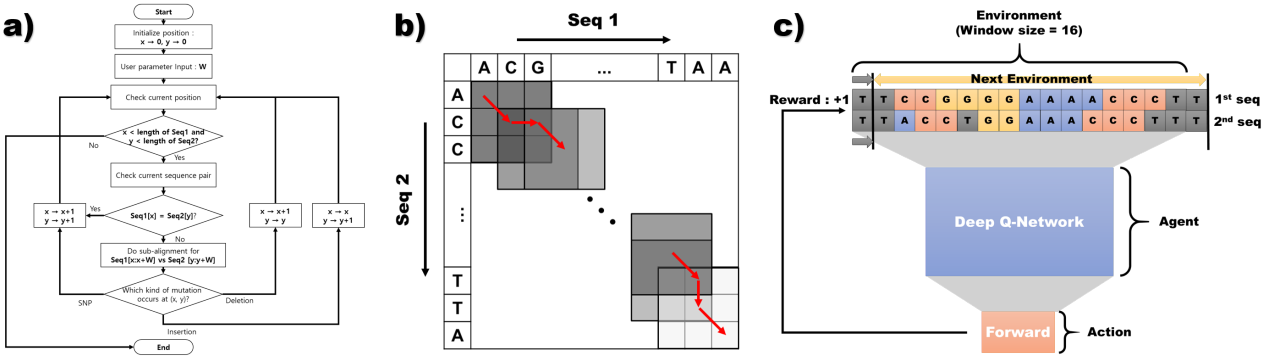

Figure 1: a) Total process of the proposed algorithm. The proposed method is divided into two parts. First, b) Local best path selection method, which is a method of repeating window movement by setting direction of progress through alignment process between small sub-sequences. Second, c) deep reinforcement learning based local best path selection method which solves high complexity problem of repeating small alignments. Then, reinforcement learning defines subsequences of current window as environment, and agent is defined as deep reinforcement learning. Here, reinforcement learning proceeds based on rewarding according to scoring strategy of sequence alignment.

The key contribution of this paper is as follows:

- Define the agent and environment for applying the deep reinforcement learning algorithm to the sequence alignment system

- Investigate the effect of each parameter to the performance of sequence alignment

- Combine the conventional sequence alignment algorithm and deep reinforcement learning based extension algorithm to improve the coverage of global sequence alignment

- Prove how the proposed algorithm can achieve the identical results with optimal global alignment

- Suggest a novel approach to the sequence alignment method with broad scalabilty

2. Materials and Methods

A entire procedure of the DQNalign is described as the flow chart in the Fig.1a. The detailed explanation and the code implementation of the DQNalign algorithm are available at https://github.com/syjqkr

2.1. Deep reinforcement learning in sequence alignment system

In order to apply deep reinforcement learning to sequence alignment, we tried to find a novel sequence alignment method in the proper form of reinforcement learning. Instead of observing the entire sequence at once, we propose a novel heuristic sequence alignment method that repeats the small alignment while moving the window of sub-sequence pairs. Through this method, it was possible to solve the memory problems and the method has complete linear complexity.

The proposed heuristic sequence alignment method can be expressed as shown in the Fig.1b. The problem of determining the optimal direction in sub-sequence within a window can also be seen as a kind of sub-alignment process, which can also be inferred that if the window size is expanded to the entire sequence length, then we can get the same results of the optimal sequence alignment. And we proved this relation of window size and performance with a numerical analysis, and we will discuss about that in the result section.

So, we proposed a novel deep reinforcement learning based sequence alignment system like the Fig.1c. Being set the 2 sub-sequences in the window as environment, the agent with the deep Q-network observes the current environment and selects the direction (forward, insertion, deletion) as the next action, then scoring system of conventional alignment method will be used as reward in reinforcement learning. Through this type of definition, we can learn a deep reinforcement learning based agent that can find the optimal path of the alignment at a given position.

2.2. Detailed network architecture

The detail of the deep Q-network based agent in the Fig.**??**, we used various techniques and networks of the deep reinforcement learning. The detailed methods will be dealt in this section.

2.2.1. Dueling Double Deep Q-network

First of all, we applied Dueling Deep Q-network and Double Deep Q-network methods to improve the convergence and stability of Deep Q-network (DQN). Dueling DQN is a method that divides the predicted reward (Q value) into a kind of average and variance, and each of them is called as "Value" and "Advantage", respectively. With this method, the agent can learn the scores

Table 1
Parameter count of each network architecture

Window size	DDDQN	Separable conv.
10		
30		
50		
100		

of the states and actions separately, and it helps the convergence of the learning progress. Moreover, Double DQN method is that uses duplicated network which called as target network. This target network is used for updating the main network while evading the overestimation. Frequent update of the DQN can redound difficult to escape the local minimum for the environment that is lost due to sudden change of policy. In order to solve this problem, the double Deep Q-network method uses the slow target network for prevents the policy falls into the local minimum. We use a double DQN method by converging the target network by a ratio of tau times the main network. With these techniques, finally we defined a Dueling Double Deep Q-network as **??**. A variable window sizes are used as parameter of the convolutional neural networks. Each of the type of nucleotide (A,C,G,T) was converted as a 3x3 pixel square with CMYK color. To separate the left, right, top, and bottom end of the sub-sequences, 3x3 pixels of empty space were added. The detailed parameters of the network is shown in the Fig.**??**. By defining the fixed filter size, the number of parameters and FLOPS were used to be linearly proportional to the size of the window size as shown in the the Tab.1. Finally, we also used the experience replay methods to prevent the overestimation.

2.2.2. Separable convolutional layer based acceleration

In order to improve the computational complexity of the proposed network, we defined a second version of the neural network using a separable convolutional layer. Separable convolutional layer separates a convolutional layer into two different layer which called point-wise layer and depth-wise layer. As shown in the Fig.**??**, this method reduces the number of calculations required for the entire convolutional layer from to . Then, the modified version of proposed network is

shown in the Fig.**??**. Similarly, ACGT were mapped into CMYK, and an empty space was applied to the edge. However, at this time, each nucleotide was mapped to a smaller size of 2x2 pixels. In addition, unlike the previous network, the number of layer was 4. At this time, the size of all filters had size of 3x3, and the stride had 3, 1, 1, and 3. Also, the maxpooling layer was added to the layer to continuously reduce the size of the layer. Detailed parameters are noted at the Fig.**??**. The changed complexity is calculated in the Table.1. With separable convolutional layer, we can reduce the number of the operations by times compared to the previous one.

2.3. Training procedure

Most machine learning requires a massive number of the input data. In addition, limited number of actual sequences can cause a bias of the deep reinforcement learning. Therefore, we had generated a various series of sequence set through the model of evolution. First, we generated one complete random sequence. Then, we made the other sequence by mutating the original one. For convenience, we used the JC69 model to create the SNP mutations [**??**]. Also, we used the zipfian distribution based indel length model for generating the indels [**??**].

2.4. Preprocessing methods of conventional sequence alignment algorithms

Prior to applying the proposed heuristic sequence alignment method, the starting point of alignment had a great influence on the result. So, we brought the several preprocessing process of the conventional alignment algorithms to show a wide scalability of the proposed method.

2.4.1. Longest common substring

As the method that can clearly show the performance of this proposed method, the longest common substring method was first attempted. We extends the sequence alignment with the proposed method from only one longest common substring in short sequences, such as HEVs.

2.4.2. Clustal Omega

We tried to improve the performance by grafting the proposed method to the pairwise alignment method used by the Clustal Omega as a global alignment tool. We refered to the code implementation of clustal omega at . The method of Clustal Omega quickly checks the k-tuple matches of

the two sequences, and record the diagonal position of each match. Thereafter, the score of each diagonal is calculated, and k-tuple matches that are located at diagonal positions as close to the window as the diagonals of the upper scores are used. However, the connection between the diagonals could be too wide to link when aligning large sequences. In this study, we aim to enable higher performance alignment by connecting these empty areas of the alignment through the proposed algorithm.

3. Results

To show the feasibility and performance of the proposed algorithm, we designed the following 3 simulations: 1. Numerical analysis on the step error probability, 2. Performance difference on the change of the parameters and 3.Performance comparison with the conventional alignment methods. With these simulations, we want to show how the proposed method can improve the alignment performance and how can we adopt the proposed method into the conventional alignment methods.

3.1. Numerical results

Before applying the deep reinforcement learning, it is necessary to prove clearly whether the heuristic sequence alignment method in Fig.1 can perform properly. So, we refers the paper [**??**] to analyze the step error probability of the proposed heuristic alignment algorithm. In this paper, the distribution of the alignment scores follows the Gumbel distribution of the multiplication of two sequence lengths like the Eq.1. With this result, we proved that the proposed method can perform like an optimal alignment in large window size.

$$P(S(A, B) \leq s) \sim \exp(-Kmne^{-\lambda s}) \tag{1}$$

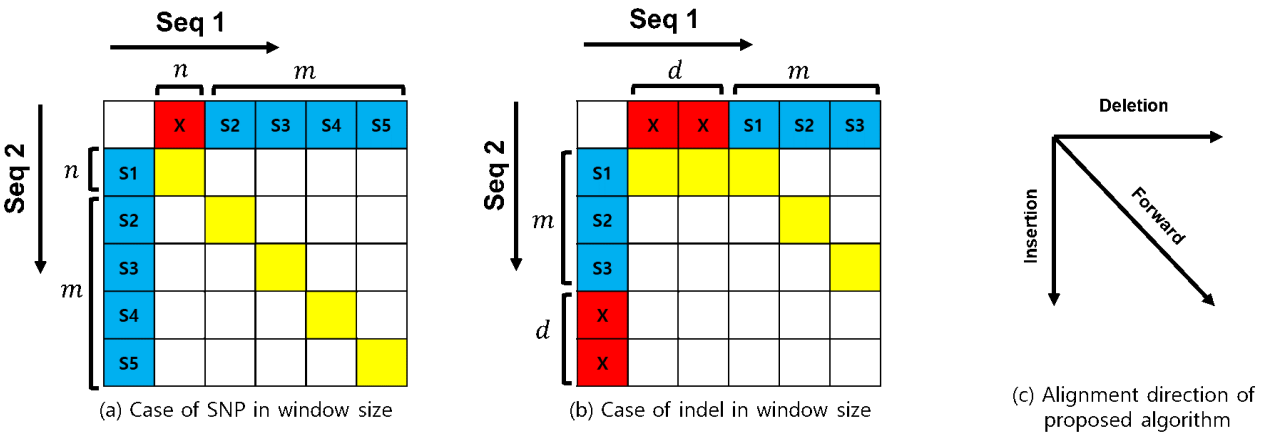

Figure 2: Modeling of the local best path selection model for analysis of error probability in case of a fixed window size. a) Expression of SNP within window size. b) Expression of an indel within window size. c) The meaning of direction in the local best path selection model

3.1.1. Numerical analysis on the step error probability

Prior to the analysis, we made three major assumptions and had one constraint. First, it was assumed that the NW algorithm can precisely match the mutation information of the actual sequences. Second, two sequence alignments with the same score confirm that both are correct. Third, the alignment at each step in the proposed algorithm can be calculated independently. Conversely, for the constraint, we considered the alignment scoring parameters to prevent the indel preferences. Next, we could derive equations on the error probability in case of similar sequences that have positive alignment scores. At this time, we calculated the step error probability of the proposed algorithm by dividing the SNP and the indel cases. The case of the SNP occurence is depicted in the Fig.2a. Let's say that the score of the optimal alignment is $score_{ans}$. In this case, we can say that an error occurs when the score of indel direction is higher than the $score_{ans}$. Then, the step error probability can be expressed using the Gumbel distribution as follows.

$$\begin{aligned} &P(S(W, W-1) + score_{gap} > score_{ans}) \\ &\simeq 1 - \exp(-KW(W-1)e^{\lambda(score_{ans} - score_{gap})}) \end{aligned} \tag{2}$$

In this equation, the $score_{ans}$ can be expressed as $Wscore_{avg}$. Then, the final formula can be summarized for infinitely large W as follow. Also, we could confirm that the error probability converges to 0 when W is infinitely large. The detailed process is summarized in the supplementary material S2.

$$P_{e,SNP} \simeq \lim_{W\to\infty} 2Ke^{\lambda score_{gap}} \frac{W^2}{e^{\lambda W score_{avg}}} \to 0 \tag{3}$$

With the similar process, we can analyze the step error probability for the indel environment as shown in the Fig.2b.

$$\begin{aligned} P_{e,indel} \simeq & K(e^{\lambda score_{gap}} + \frac{1}{4}e^{\lambda score_{match}} + \frac{3}{4}e^{\lambda score_{mismatch}}) \\ & \times \lim_{W\to\infty} \frac{W^2}{e^{\lambda W score_{avg}}} \to 0 \end{aligned} \tag{4}$$

Finally, the final step error probability reflecting each indel occurrence probability and SNP occurrence probability is as follows.

$$\begin{aligned} P_{e,total} \simeq & (p_{indel}K(e^{\lambda score_{gap}} + \frac{1}{4}e^{\lambda score_{match}} + \frac{3}{4}e^{\lambda score_{mismatch}}) \\ & + 2p_{SNP}Ke^{\lambda score_{gap}}) \frac{W^2}{e^{\lambda W score_{avg}}} \to 0 \end{aligned} \tag{5}$$

3.1.2. Simulation results to the step error probability analysis

To verify the step error probability equation through numerical analysis, we designed a simple simulation. In the in-silico simulation, we considered the SNP and indel probability between two sequences like the environment table in the Tab.**??**. Then, we can get the results shown in the Fig.**??**. Through this simulation, we were able to confirm that the tendency in the simulation and the equation are similar. And in the high window size, the step error rate converged to near zero.

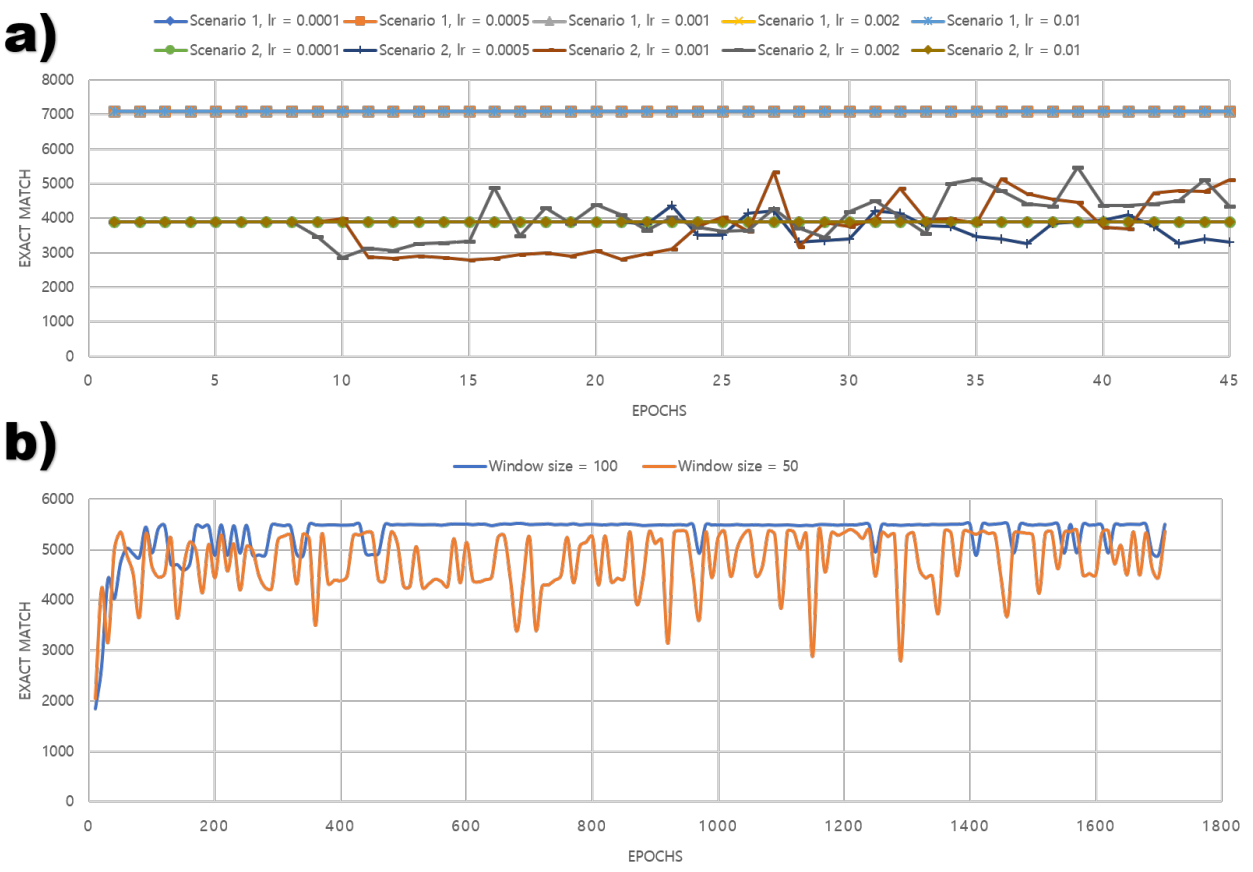

Figure 3: Performance convergence graph of the proposed method. a) The tendency changes of the convergence with various learning rates in the case that the window size is 100. b) The part of the convergence graph in the total training procedure

3.2. Simulation results with the proposed method

We designed the following three simulations to compare the performance of various parameters in the proposed scheme: 1. performance convergence of the deep reinforcement learning-based alignment in training procedure, 2. performance comparison on various parameters. With these simulations, we intend to show the process of adapting deep reinforcement learning to alignment, along with consideration of the optimal parameters of the proposed method.

3.2.1. Performance convergence in the training procedure

The result of the performance convergence of the proposed method in the training procedure is shown in the following Fig.3.

In the training procedure, it was confirmed that the case of scenario 1 in the Tab.**??**, which is a scenario of alignment of two similar sequences, converges with a very short time regardless of the learning rates. This is a phenomenon that occurs because SNP and match cases are dominant, and it is a result given by AI learning easily that high scores can be obtained only by judging in the forward direction regardless of the environment.

However, in order to proceed with the actual sequence alignment, it is necessary to learn in various environments other than these biased results. The Fig.3a shows the learning results in a difficult environment, the case of scenario 2 in the Tab.**??**. As can be seen in the result, it did not show the change of the results in some of the learning rates, and showed the agent can escape the biased results within a proper learning rate. Therefore, we could figure that it is important to tuning the learning rates along with the change of the other parameters. Then, the part of the convergence graph is depicted in the Fig.3b. Like the results in this figure, by adjusting the learning rates, we could converge the networks in thousands of epochs. And finally we used converged networks in the later parts of the results. Also, you can download the converged network in our github link.

3.2.2. Effect of the parameters

To observe the effect of the parameters, we used the simulation environments in the Tab.**??**. At this time, the number of the exact matches are used to compare the performance of the proposed algorithm. Also, we considered the Needleman-Wunsch algorithm's result to know the ideal result of the each simulation cases. Then, the result of sequence alignment in the actual sequence is shown in the Fig.4.

As can be seen in the Fig.4, an increase of the window size resulted in an increase in performance. When the window size was over 100, the performance was converged and showed almost the same result. Therefore, we concluded that 100 is the optimal window size for the HEV sequence set.

In addition, the proposed method is not limited to a specific neural network structure which means that it can be learned using various network structures. As can be seen in the figure, it was

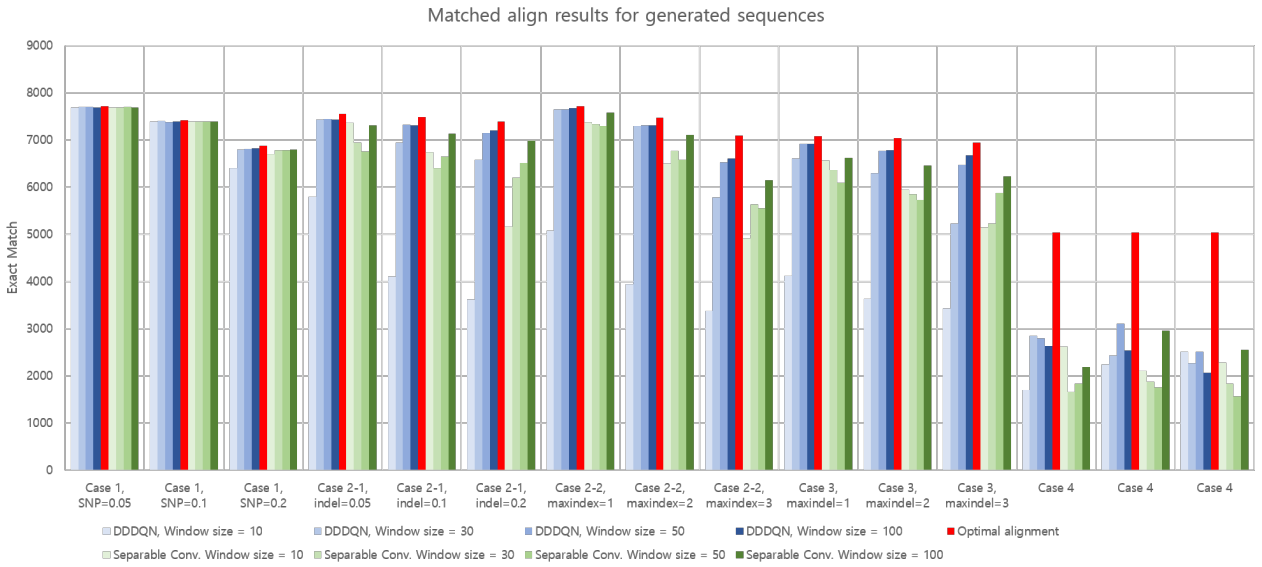

Figure 4: Exact match results of proposed method and the optimum results in various simulation scenarios

Table 2
Average time spent in simulation with various networks (s)

Window size	10	30	50	100
DDDQN	1.916	2.910	2.522	2.911
Separable conv.	1.080	1.327	1.302	1.411
Optimal alignment	143.6	-	-	-

confirmed that the normal DDDQN structure has slightly better performance than the seperable conv structure. Despite this, in the complexity results in the Tab.2, it was confirmed that the execution time is about 2 times faster than the general DDDQN structure. Based on these results, we were able to check the pros and cons of the alignment algorithm using each of networks. Also, we used the optimal window size for each network to 100 in the next section for treating the actual sequence data.

3.3. Comparison with conventional alignment algorithms

In order to confirm the difference between the proposed method and the conventional sequence alignment algorithm, two sequence sets were used: 1. HEV genome sequence set, 2. two E.coli

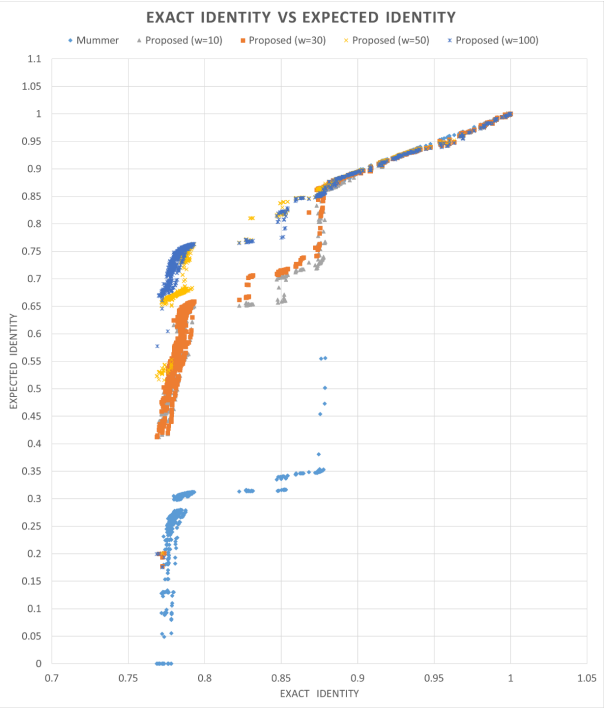

Figure 5: Exact match results in real sequences, which consist of the 47 Hepatitis E Viruses. Thus, in this sequence set, there are 1081 pairs.

genome sequences. In the rest of this part, we compared the exact match performance and time complexity between various sequence alignment methods: Clustal omega, MUMmer and the proposed method.

3.3.1. Simulations on the HEV genome sequence set

The results of the HEV sequence set in the Tab.**??** are shown in the Fig.5. All the alignment results are attached in the supplementary material S2. At this time, the ratio of exact matches against the performance of optimal alignment was used for performance measure. As shown in the Fig.5, it was confirmed that not only the proposed method, but also the conventional methods have similar performances for sequence sets which has 0.88 or higher identity value.

However, in the lower identity range, the difference within the alignment methods began to increase. In the case of the proposed method, it was confirmed that both the DDDQN and separable convolutional layer based networks can maintain the similar alignment performance as the optimal sequence alignment method in over 78% identity value.

Table 3
Alignment results of the E.coli sequences: Escherichia coli O157:H7 str. TW14359 vs. Escherichia coli str. K-12 substr. MG1655

	Mummer	Only DQNalign	Only Clustal	Clustal + DQNalign
Exact matches		2359437	3890193	4088757
Consumed time (s)	10.81	35133	514139	514139 + 32566

The main reason of difference is that the case of dissimilar sequence pair, the number of anchors (Ex. k-tuples in the Clustal method and MUMs in the MUMmer) rapidly decreases, and it causes the failure in the process of connecting the anchors. So, the entire alignment cannot be continued and the conventional alignment methods could achieve low coverage and exact matches. However, the proposed methods immediately observes the window and decides the direction of alignment regardless of the anchors. So, the proposed method had a less difficulty in aligning the sequence pairs which has relatively low identity values.

3.3.2. Simulations on the E.coli genome sequence set

Finally, experiments on Escherichia coli O157:H7 str. TW14359 and Escherichia coli str. K-12 substr. MG1655 were also proceed. As expected, sequence alignment with E.coli was not easy to our proposed method. It is because there are large genomic variations with the gene scale in the E.coli samples, it was hard to connect the whole sequences using only a few hundred-sized windows. Here, the whole genome alignment result was obtained through fusion with Clustal omega's method. The detailed parameters are written in supplementary material S2, and the alignment results are stored in supplementary material S3.

As shown in Tab.3, the exact match result of the proposed method was lower than the conventional methods because of the insufficient pre-processing. So, we decided to graft the pre-processing method of the clustal method. We used the proposed method to align unaligned gaps in the alignment result of the clustal method. With this combination, we could align the whole E.coli sequences properly.

In addition, we can see that the time required for additional expansion with the proposed method

is much shorter than the pre-processing procedure. Compared with Mummer, it was very slow, but in the case of the proposed method, we could see the possibility of proceeding within a few hours to align long sequences over 5Mbp. Also, we could confirm that the proposed method is also suitable to integrate with the conventional method.

4. Discussion and Conclusion

In this study, we proposed a deep reinforcementa learning based alignment method to reduce the complexity of pairwise alignment. Using only simple preprocessing based on the longest common substring (LCS), the proposed algorithm can complete the entire sequence alignment by repeating small alignments. Through the analysis, we have confirmed that the proposed method achieves high accuracy and low complexity by adjusting the window size adaptively. In addition, we revealed the advantages and disadvantages of the proposed method by analyzing the generated and actual sequences. We observed that the proposed algorithm has more accurate alignment performance in a wider range of sequences than the conventional methods when a proper window size is used. However, we confirmed that the MUMmer method has a much faster alignment performance for generated and real genome sequences. Nevertheless, it was revealed that sequences with millions of bases, such as E. coli, could be aligned within a few hours by combining conventional alignment algorithm and the proposed alignment algorithm.

In this study, we confirmed that the proposed method has the possibility of aligning two long sequences with less complexity. Since the proposed algorithm used deep reinforcement laerning based selection rather than human-based features, the alignment result was not stable for some sequence pairs. Therefore, we intend to strengthen the proposed method by adopting ensemble methods. Also, there were several challenges in securing performance on actual sequences, which result from the gap between the modeled sequence and the actual sequence used in the learning. We intend to solve these issues in future by using a learning method that directly reflects the actual sequences. In addition, we will attempt to apply the proposed method to local alignment and multiple sequence alignment with improved learning methods and training strategies.

References

[1] Schuster, Stephan C, “Next-generation sequencing transforms today’s biology” *Nature methods,* vol. 5, no. 1, pp. 16, Dec. 2007.

[2] Needleman, Saul B., and Christian 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.

[3] Edgar, Robert C, "MUSCLE: multiple sequence alignment with high accuracy and high throughput" *Nucleic acids research,* vol. 32, no. 5, pp. 1792-1797, Mar. 2004.

[4] Katoh, Kazutaka, and Daron M. Standley, "MAFFT multiple sequence alignment software version 7: improvements in performance and usability" *Molecular biology and evolution,* vol. 30, no. 4, pp. 772-780, Jan. 2013.

[5] Sievers, Fabian, et al, "Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega" *Molecular systems biology,* vol. 7, no. 1 pp. 539, Oct. 2011.

[6] Wang, Lusheng, and Tao Jiang, “On the complexity of multiple sequence alignments,” *Journal of computational biology,* vol. 1, no. 4, pp. 337-348, Jan. 1994.

[7] Chao, Kun-Mao, William R. Pearson, and Webb Miller, “Aligning two sequences within a specified diagonal band,” *Bioinformatics,* vol. 8, no. 5, pp. 481-487, Oct. 1992.

[8] Camacho, Christiam, et al, “BLAST+: architecture and applications,” *BMC bioinformatics,* vol. 10, no. 1, pp. 421, Dec. 2009.

[9] Marçais, Guillaume, et al, “MUMmer4: A fast and versatile genome alignment system,” *PLoS computational biology,* vol. 14, no. 1, pp. e1005944, Jan. 2018.

[10] Mnih, Volodymyr, et al, “Playing atari with deep reinforcement learning,” *arXiv preprint,* vol. 1312, no. 5602, pp. 1, Dec. 2013.

[11] Van Hasselt, Hado, Arthur Guez, and David Silver, “Deep reinforcement learning with double q-learning,” in *Thirtieth AAAI conference on artificial intelligence 2016.*

[12] Wang, Ziyu, et al, “Dueling network architectures for deep reinforcement learning,” *arXiv preprint ,* vol. 1511, no. 06581, pp. 1, Apr. 2016.

[13] Mott, Richard, “Maximum-likelihood estimation of the statistical distribution of Smith-Waterman local sequence similarity scores,” *Bulletin of Mathematical Biology,* vol. 54, no. 1, pp. 59-75, Jan. 1992.

[14] Pang, Hongxia, et al, “Statistical distributions of optimal global alignment scores of random protein sequences,” *BMC bioinformatics,* vol. 6, no. 1, pp. 257, Oct. 2005.

[15] Tang, Jie, et al, “A novel k-word relative measure for sequence comparison,” *Computational Biology and Chemistry,* vol. 53, no. 1, pp. 331-338, Dec. 2014.

[16] Hayashi, Tetsuya, et al, “Complete genome sequence of enterohemorrhagic Eschelichia coli O157: H7 and genomic comparison with a laboratory strain K-12,” *DNA research,* vol. 8, no. 1, pp. 11-22, Feb. 2001.

[17] Jukes, Thomas H., and Charles R. Cantor, “Evolution of protein molecules,” *." Mammalian protein metabolism,* vol. 3, no. 21, pp. 132, Sep. 1969.

[18] Qian, Bin, and Richard A. Goldstein, “Distribution of indel lengths,” *Proteins: Structure, Function, and Bioinformatics,* vol. 45, no. 1, pp. 102-104, Aug. 2001.

[19] Matsubara, Wataru, et al, “Efficient algorithms to compute compressed longest common substrings and compressed palindromes,” *Theoretical Computer Science,* vol. 410, no. 8-10, pp. 900-913, Mar. 2009.