A High Precision Heuristic for Motif Extraction Using Random Forests

Jigen Murata, Masato Inagi, Martin Lukac, Shin'ichi Wakabayashi, Shinobu Nagayama
Department of Information Engineering, Graduate School of Information Sciences,
Hiroshima City University, Hiroshima, Japan
mj66020@e.hiroshima-cu.ac.jp, {inagi, malu, wakaba, s_naga}@hiroshima-cu.ac.jp

Abstract— The motif extraction is a problem to find similar substrings, called motifs, from many DNA sequences. The Gibbs sampling is the best known heuristic for this problem, but it has a disadvantage that it tends to fall into a local optimum because solution search space is limited due to its greedy search. This paper proposes a heuristic that uses Random Forests to classify substrings and performs a wide range of solution search. Since the proposed heuristic improves precision of solutions significantly, this paper also proposes a hybrid method combining the Gibbs sampling with the Random Forest based method to reduce computation time while keeping the precision of solutions.

I. Introduction

In the field of bioinformatics, it is assumed that common or similar substrings, called motifs, among DNA sequences have same or similar characteristics, structures, and functional roles. Thus, finding such motifs is an essential problem in applications such as biological classification and virus antibody detection. This problem is also known as the motif extraction [1]. Since the number of DNA sequences and their length are large, a method finding motifs quickly and precisely has been required.

The motif extraction belongs to the class of NP-hard problems and is too hard to solve the problem by exact solution methods. Therefore, heuristic methods are necessary, and the Gibbs sampling (GS) [2] is the best known heuristic for this problem. The GS can find a solution with a certain degree of precision in a short time. On the other hand, it has a disadvantage that it tends to fall into a local solution because solution search space is limited due to its greedy search. However, improving the precision of solutions is hard, and thus, it has been hard to beat the GS. Although there exist its variants in implementation [1, 3, 4, 5], its algorithm remains basically unchanged.

To make a new breakthrough, this paper proposes a heuristic method using Random Forests (RFs), an ensemble machine learning, for the motif extraction. This is because their learning speed is fast, and RFs can efficiently classify data without labels (i.e., unsupervised learning is possible). In this paper, the method using RFs is called Random Forest motif search (RFMS). The RFMS uses RFs to classify similar substrings in given DNA sequences in advance. Thereby, it efficiently narrows down candidates and extracts promising substrings. Experimental results show that the proposed heuristic method improves precision of solutions significantly, compared to the existing GS, in a reasonable computation time.

The RFMS obtains higher precision solutions, but has a disadvantage requiring longer computation time than the GS. To reduce computation time while keeping the precision of solutions, this paper also proposes a hybrid method combining the GS with the RFMS. Additional experimental results show that the hybrid method takes advantages of the GS and the RFMS, and it is promising.

II. Preliminaries

A. Formulation of Motif Extraction

This section formulates the motif extraction. A motif is defined as a set of substrings common or similar among the given biological sequences, such as DNA sequences or amino acid sequences. Fig. 1 shows an example of a motif. To evaluate the similarity among substrings, Equation (1) is often used.

$$E = \frac{1}{L}\sum_{j=1}^{L}\sum_{\omega\in\Omega} f_j(\omega)\log_2\frac{f_j(\omega)}{p(\omega)} \tag{1}$$

In Equation (1), Ω is the set of characters composing biological sequences. $p(\omega)$ is the background probability of the character ω that ω appears in the set of biological sequences. Similarly, $f_j(\omega)$ is the occurrence probability of the character ω that ω appears at the j-th position in the set of extracted substrings with the length L.

The motif extraction is formalized as follows: Given a set of biological sequences $S = \{s_1, s_2, ..., s_n\}$ and the length L of motif to be extracted, find a motif $M = \{m_1, m_2, ..., m_n\}$ maximizing Equation (1), where the length of each substring m_i is L. For simplicity, this paper assumes that the length of all the given sequences is equal, and they have no gaps. We also assume that the given sequences are DNA sequences, and thus, the character set $\Omega = \{A, T, G, C\}$.

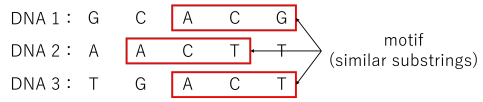

Fig. 1. Example of a motif

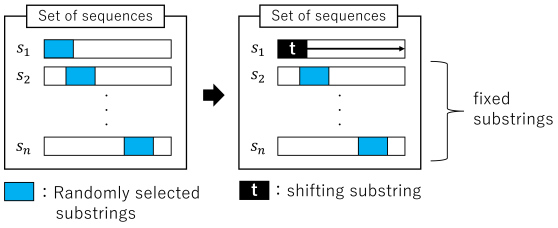

Fig. 2. Overview of the Gibbs sampling

B. Gibbs Sampling

B.1. Algorithm

The GS begins with randomly selecting a set of substrings of length L from given sequences as an initial solution. Then, it heuristically searches for the set of substrings maximizing Equation (1) while updating the solution by shifting each substring in a sequence in turn. The same process is iterated for a specified number of times. A part of the overview of the GS is shown in Fig. 2.

The following is the process flow of the GS.

1. Randomly select a substring m_i of length L from each sequence as an initial solution ($i = 1, 2, \ldots, s_n$).
2. For each sequence s_i, iterate the Steps 3 to 5 for a specified number of times U.
3. For each position from the 1st to the $(|s_i| - L + 1)$th in s_i, extract a substring and compute Equation (1) along with other fixed substrings of other sequences, where $|s_i|$ is the length of sequence s_i.
4. Among substrings extracted in the Step 3, let the substring maximizing (1) be a new candidate motif m_i in s_i.
5. If the value of (1) does not change for a certain number of iterations before reaching the specified number of times U, proceed to the Step 6.
6. Output the obtained motif M and the value of (1), then terminate.

B.2. Features of the Gibbs Sampling

The GS can find a motif heuristically even from a huge number of long biological sequences in a short time. However, it tends to fall into a local optimum or not to converge sufficiently with respect to precision of solution. This is because its search space of solutions is limited due to its greedy search.

As shown in the previous subsection, the GS greedily searches for solutions based on Equation (1) without allowing any changes to worse solutions, and it searches by shifting only one substring in a sequence while fixing others. Thus, its search space of solutions tends to be limited. To overcome the drawbacks, some studies [4][5] have been reported. Although they improve the precision of solutions by re-selecting initial solutions or enforcedly changing solutions to climb out of a local optimum, they do not still reach a breakthrough. Therefore, more effective methods are required, and searching a wider solution space is still regarded as a potentially beneficial approach to achieve better results than GS.

This paper proposes a method using RFs as a completely different approach than previous ones.

C. Random Forest

RF [6][7][8][9][10][11] is a supervised ensemble machine learning in which multiple weak learners are combined in parallel to form a strong learner. A RF consists of multiple decision trees (DTs), and each DT is trained independently.

The learning and inference of RFs are shown in Figs. 3 and 4, respectively. In the learning phase, a DT is constructed from each subset of the given dataset, as shown in Fig. 3. A subset is randomly sampled from the given dataset with correct labels. Each subset is then partitioned recursively by each node using an attribute of data and its threshold value into smaller subsets. Finally, leaf nodes of a DT has the partitioned subsets with labels. The same process is repeated n times to construct n DTs. It is known that by selecting subsets and attributes randomly, DTs are independent each other, resulting in high quality of results [12].

In the inference phase, as shown in Fig. 4, the constructed DTs are used to predict the result for unknown data (data without labels). According to attributes of the unknown data, all the DTs are traversed by selecting an appropriate edge at each node from root nodes to leaf nodes. Then, the final result is predicted by majority voting among the labels extracted from the reached leaf nodes (in Fig. 4, the result is "A"). In general, majority voting is used for classification tasks, and an average is taken for regression tasks.

III. Proposed Heuristic

A. Random Forest Motif Search (RFMS) Method

A.1. Overview of RFMS

To overcome the disadvantage of the GS, it is necessary to search wider space of solutions efficiently. Therefore, RFMS uses RFs to coarsely classify similar substrings in

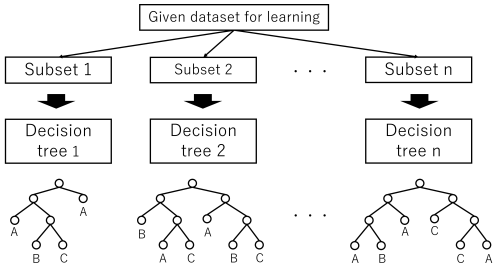

Fig. 3. Learning of Random Forest (constructing trees)

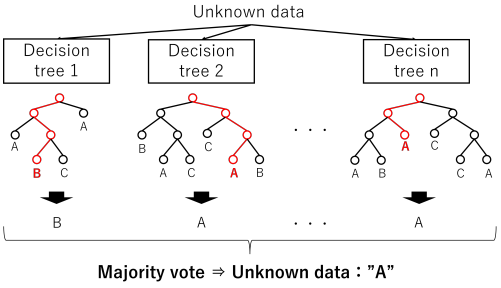

Fig. 4. Inference of Random Forest (traversing trees)

biological sequences in advance. Then, it finds the most similar substring in each sequence among classified substrings. Fig. 5 shows an overview of the RFMS. In the learning phase, a RF is constructed for each sequence s_i that classifies substrings randomly sampled from each sequence s_i. In the inference phase, for each substring in s_i, similar substrings in other sequences than s_i are searched in parallel using their RFs constructed in the learning phase. Among the obtained sets of substrings, one maximizing Equation (1) is selected as the motif, and then the selected set is output. The following subsections describe each phase in detail.

A.2. Learning Phase

As described in A.1, in the learning phase, substrings are randomly sampled from each sequence s_i and are classified to construct a RF for each s_i. To construct a DT in a RF, the set of sampled substrings is recursively divided into

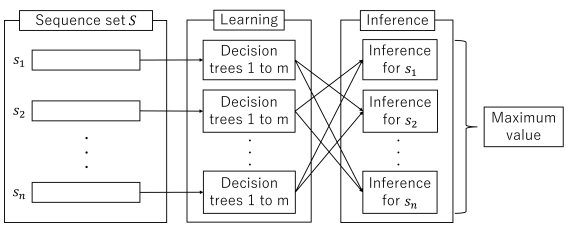

Fig. 5. Overview of RFMS

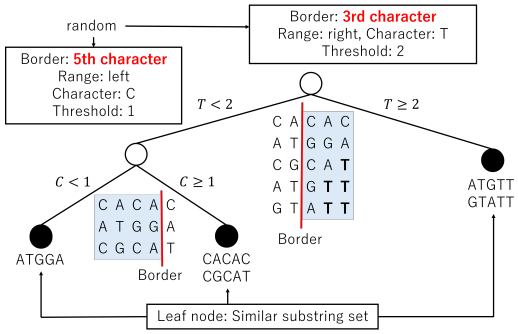

Fig. 6. Classification of substrings by a decision tree

two similarly sized subsets at each node of the DT by a randomly selected attribute (feature) of substrings and its threshold value. In this paper, the attribute is the number of characters contained within a specified range of substrings.

Fig. 6 shows an example of a DT classifying 5 substrings: "CACAC","ATGGA","CGCAT","ATGTT" and "GTATT". At the root node, the 3rd character as the border, the right of this border as the range, 'T' as the character, and 2 as the threshold are selected, and they are used to divide the substrings. If the number of occurrences of the character "T" within this range is smaller than the threshold of 2, the substrings are classified into the left node; otherwise, they are classified into the right node. Similarly, "CACAC","ATGGA","CGCAT", which are classified into the left node, are classified again by the 5th character, the left of the border, 'C', and 1. After constructing the DT, sets of coarsely classified substrings are obtained at leaf nodes. Note that substrings at leaf nodes have no labels. That means this is a unsupervised learning.

The following is the algorithm for the learning phase.

1. Repeat the Step 2 for each sequence s_i (i=1,2,...,n).
2. Repeat the following steps a to c to create m DTs.
   a. Randomly extract r substrings of length L from s_i and let the set of substrings be p_x $(x = 1, 2, \ldots, m)$
   b. Randomly select l of d attributes to divide p_x, where l is the floor of $\sqrt{d}$.
   c. Using one of the l attributes at each node, p_x is recursively divided until either the tree depth reaches the specified number *maxdepth* or the number of substrings at leaf nodes reaches the specified number *size*.

A.3. Inference Phase

Fig. 7 shows an overview of the inference phase in the RFMS method. The RFMS begins with extracting a substring t of length L from a sequence s_i one by one. Then,

it searches for substrings similar to each t in sequences s_j's other than s_i, respectively. This is done by traversing RFs for s_j's. Fig. 8 shows the search method for a substring similar to t using a RF for s_j. In a RF, m DTs are traversed according to t, and then, the most similar substring to t among substrings classified in leaf nodes is extracted. In this way, $n-1$ substrings in s_j's similar to each t are extracted as a motif candidate.

Since Equation (1) computes the similarity among the whole set of n substrings, it should be modified as follows to compute the similarity between two substrings appeared in the above process.

$$E' = \frac{1}{L}\sum_{c=1}^{L}\sum_{\omega\in\Omega} f'_c(\omega)\log_2\frac{f'_c(\omega)}{p'(\omega)} \qquad (2)$$

The flow of the computation of E' is shown in Fig. 9. Unlike Equation (1), Equation (2) uses the background probability $p'(\omega)$ and the occurrence probability $f'_c(\omega)$ computed from only two targeting sequences (s_i and s_j) and two substrings, respectively.

The following is the algorithm for the inference.

1. Repeat the following steps a and b for each sequence s_i $(i = 1, 2, \ldots, n)$ to extract a set of n substrings that is the best solution in inference for s_i.
   (a) Repeat the following steps i and ii for each substring t_k of length L in s_i from the 1st to the $(|s_i| - L + 1)$-th.
      i. Search for the most similar substring to t_k in each of sequences s_j's other than s_i using a RF for s_j and Equation (2) to obtain a set of $n-1$ similar substrings.
      ii. Compute Equation (1) in the set of n substrings including t_k (i.e., a motif candidate).
   (b) Among the $|s_i| - L + 1$ substring sets obtained by the step (a) (repetition of the steps i and ii), select the substring set with the maximum value of (1) as the best solution for s_i.
2. Among the n substring sets, each of which is the best solution in each s_i, obtained in the above, output one with the maximum value of (1) as the solution of the algorithm.

B. *Hybrid Method*

As will be shown later, the RFMS successfully improves the precision of solutions. However, it requires a longer computation time than the GS. Although it is a reasonable computation time, faster computation would be better in various applications. To reduce computation time while keeping the precision of solutions, this section proposes a hybrid method combining the GS with the RFMS.

As shown in Section A, the RFMS can efficiently perform global search of solutions, but it does not perform

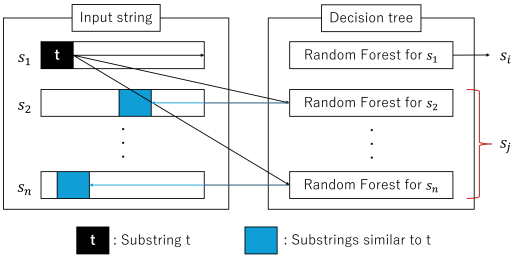

Fig. 7. Overview of inference

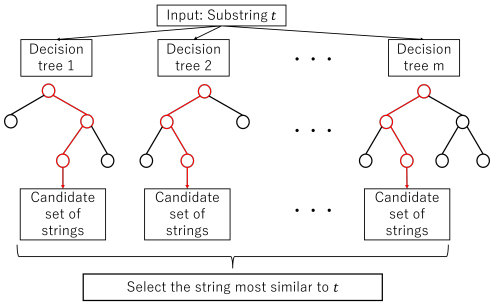

Fig. 8. Search for the most similar substring to t

local search of solutions (i.e., improvement by detailed search of neighbors). This is because searches in the RFMS are based on coarse classification of randomly sampled substrings. To cover the effects expected in local search by the RFMS, more sampled substrings and more DTs for a RF are needed, resulting in a longer computation time.

On the other hand, the GS is good at local search of solutions, although its search space of solutions is relatively narrow due to its sequential search of neighbors. Therefore, the proposed hybrid method achieves faster computation while keeping the precision of solutions by taking advantages of both search methods. Specifically, the hybrid method globally searches for a promising solution by the RFMS, and then, it improves the precision of the obtained solution greedily by the local search of the GS. Fig. 10 shows an overview of the hybrid method. As shown in Fig. 10, it begins with producing a set of substrings with the maximum value of (1) (promising so-

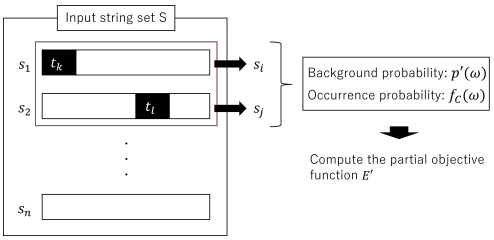

Fig. 9. Computation flow of Equation (2): E'

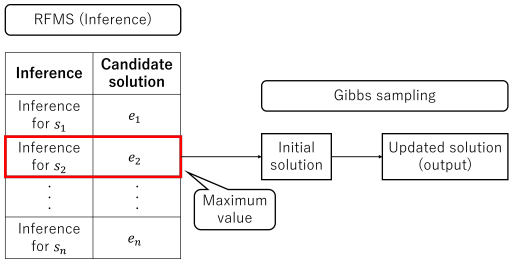

Fig. 10. Overview of hybrid method

lution) by the RFMS. Then, considering the set of substrings as an initial solution for the GS, the precision of the solution is improved by the GS.

The following is the algorithm of the hybrid method.

1. Perform the learning phase and the Step 1 of the inference phase in the RFMS to extract the best solution e_i in the inference for each s_i $(i = 1, 2, \ldots, n)$.
2. Select the solution with the maximum value of (1) among e_i's.
3. Set the selected solution as an initial solution, and then, perform the Steps 3-5 of the GS to improve it.
4. Output the improved solution as the solution of the algorithm.

IV. Experiments and Evaluations

A. Experiment Environments

In the experiments, the GS, the RFMS, and the hybrid method are implemented in C and compared. The experimental environments are as follows: CPU: AMD Ryzen 7 3700U with Radeon Vega Mobile Gfx 2.30GHz, memory: 8.00GB, C compiler: gcc (optimization option: -O2). As the experimental data, we use the Homo sapiens DNA sequence dataset (length of each sequence: 300, number of sequences: 50) [13]. To evaluate the quality of obtained solutions by the three heuristics (how far obtained ones are from the exactly optimum solution) precisely, we embedded the identical substring with length L into each sequences of the above data intentionally. The value of Equation (1) gets the maximum only when the embedded substrings are found as the motif. This means they are the exact solution. Experiments are performed 10 times for each of motif lengths $L = 5, 10, 15, 20, 25$, and 30. Section B shows the average values of (1) and average computation times in the 10 experiments.

B. Experimental Results and Discussion

B.1. Gibbs Sampling

In this experiment, the maximum number of iterations for the GS is set to $5,000$. However, as shown in the algorithm of the GS, it can terminate in even fewer iterations if no improvement of solution is obtained during several iterations. In that case, we can consider that the obtained solution falls into a local optimum. Table I shows results of the GS.

TABLE I
Experimental results of Gibbs sampling

L	Value of (1)	Exact value*	Computation time in sec.
5	1.64	2.02	3.26
10	1.23	1.99	7.09
15	1.24	2.00	10.24
20	1.36	1.99	13.31
25	1.36	1.98	17.12
30	1.43	1.98	19.69

*Exact value denotes the value of (1) for exact solution.

From Table I, we can see that the GS finds a solution quickly even when L is large. However, the difference between the value of (1) for the obtained solution and that for the exact solution is large. This means that the obtained solutions are local optimum (i.e., there is much room for improvement).

B.2. RFMS

In the experiment for the RFMS, the number of substrings to be randomly sampled for a DT is 30, the number of DTs in a RF is 25 or 50, the number of substrings at each leaf node is up to 5, and the maximum height of DTs is 7. Tables II and III show the results when the number of DTs is 25 and 50, respectively.

Table II shows that the RFMS obtains solutions closer to the exact ones than the GS at the expense of computation speed. In addition, from Table III, we can see that by using more DTs, the RFMS finds even the exact solutions. A breakdown of computation time for the RFMS shows that longer time is spent for traversing RFs during the inference phase, rather than for constructing them during the learning phase.

In this way, the RFMS improves the precision of solutions significantly in a reasonable computation time.

B.3. Hybrid Method

In the experiment for the hybrid method, parameters for the RFMS are the same as the previous experiment, and the maximum number of iterations for the GS is only 10. Table IV shows the experimental results of the hybrid method.

Table IV shows that the hybrid method obtains almost exactly optimum solutions in a computation time comparable to the RFMS with 25 trees. These good results are

TABLE II
Experimental results of RFMS with 25 trees

L	Value of (1)	Exact value	Computation time (sec.)	
			Learning	Inference
5	1.99	2.02	0.02	40.46
10	1.86	1.99	0.03	71.56
15	1.89	2.00	0.03	97.36
20	1.83	1.99	0.04	124.51
25	1.86	1.98	0.04	160.77
30	1.87	1.98	0.04	185.73

TABLE III
Experimental results of RFMS with 50 trees

L	Value of (1)	Exact value	Computation time (sec.)	
			Learning	Inference
5	2.02	2.02	0.04	89.90
10	1.99	1.99	0.05	154.35
15	2.00	2.00	0.06	216.36
20	1.99	1.99	0.07	274.73
25	1.98	1.98	0.07	330.45
30	1.98	1.98	0.07	379.41

achieved due to efficient global search by the RFMS and fast local search by the GS.

Results with similar quality might be obtained even faster by using the RFMS with fewer trees and the GS with more iterations. We believe that there exists an optimal balance between the computational loads of RFMS and GS, and our future work includes discovering such a balance.

V. Conclusion

In this paper, we proposed a new heuristic method using a machine learning, RFs, to solve the motif extraction problem. It can find high precision solutions by classifying similar substrings and searching for solutions efficiently using RFs. We also proposed a hybrid method combining it with the GS to reduce computation time while keeping the precision of solutions. Experimental results show that the proposed methods are promising to make a breakthrough.

Future works include reducing the inference time in the RFMS, tuning hyperparameters, finding the optimal balance between global search and local search, and applying the RFMS to larger practical biological data.

TABLE IV
Experimental Results of Hybrid Method with 25 trees

L	Value of (1)	Exact value	Computation time (sec.)	
			RFMS	Gibbs sampling
5	2.02	2.02	41.77	0.12
10	1.99	1.99	70.94	0.22
15	2.00	2.00	108.28	0.31
20	1.98	1.99	131.75	0.41
25	1.97	1.98	160.74	0.51
30	1.97	1.98	189.34	0.61

ACKNOWLEDGMENTS: This research was partly supported by the JSPS KAKENHI Grant (C), No.23K11038, 2025.

References

[1] Y. Takahashi, H. Kitakami, S. Fukumoto, Y. Mori, K. Tamura, "Method for high-precision motif extraction based on Gibbs sampler in amino acid sequences," *Technical Report of Bioinformatics (BIO)*, 2016-BIO-46(15), pp.1-10, 2016 (in Japanese).

[2] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Lin, A. F. Neuwald, and J. C. Wootton, "Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment," *Science*, New Series, Volume 262, Issue 5131, pp.208-214, 1993.

[3] T. Akutsu, H. Arimura, and S. Shimozono, "On approximation algorithms for local multiple alignment," *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, pp.1-7, 2000.

[4] N. Kono, K. Tamura, Y. Mori, and H. Kitakami, "GS Optimization Technique for Extracting Similar Partial Sequence from Sequence Database," *IPSJ SIG MPS technical reports*, Vol.2009-MPS-76, No.46, pp.1-8, 2009 (in Japanese).

[5] Y. Yuasa, S. Nagayama, M. Inagi, and S. Wakabayashi, "A hybrid method using Monte-Carlo tree search and Gibbs sampling method for solving motif extraction problems," *IEICE technical reports*, Vol.118, No.334, VLD2018-61, pp.149-154, 2018 (in Japanese).

[6] L. Breiman, "Bagging predictors," *Machine Learning*, vol.24, no.2, pp.123-140, 1996.

[7] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.

[8] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol.1, pp.81-106, 1986.

[9] L. Breiman,"Random forests," *Machine Learning*, vol.45, pp.5-32, 2001.

[10] T. K. Ho, "Random decision forests," *Proceedings of 3rd International Conference on Document Analysis and Recognition*, pp.278-282, 1995.

[11] B. Xue, S. Nagayama, M. Inagi, and S. Wakabayashi, "A programmable architecture based on vectorized EVBDDs for network intrusion detection using random forests," *The 2017 International Symposium on Nonlinear Theory and Its Applications*, Proc. pp.132-135, 2017.

[12] M. Taguri and J. Wang, *Introduction to Data Science*, Ohmsha, 2022 (in Japanese).

[13] `https://www.ncbi.nlm.nih.gov`(accessed 2025-3)