IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 11, 2021, accepted July 15, 2021, date of publication July 19, 2021, date of current version July 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3098340

Sequence Alignment Algorithm for Statistical Similarity Assessment

JAKUB NIKONOWICZ, ŁUKASZ MATUSZEWSKI, AND PAWEL KUBCZAK
Faculty of Computing and Telecommunications, Poznań University of Technology, 60-965 Poznań, Poland

Corresponding author: Jakub Nikonowicz (jakub.nikonowicz@put.poznan.pl)

This work was supported by the Polish Ministry of Science and Higher Education, in 2021, under Grant 0314/SBAD/0210.

ABSTRACT This paper presents a new approach to statistical similarity assessment based on sequence alignment. The algorithm performs mutual matching of two random sequences by successively searching for common elements and by applying sequence breaks to matchless elements in the function of exponential cost. As a result, sequences varying significantly generate a high-cost alignment, while for low-cost sequences the introduced interruptions allow inferring the nature of sequences dependence. The most important advantage of the algorithm is an easy interpretation of the obtained results based on two parameters: stretch ratio and stretch cost. The operation of the method has been simulation tested and verified with the use of real data obtained from hardware random number generators. The proposed solution ensures simple implementation enabling the integration of hardware solutions, and operation based on only two sequences of any length predisposes the method to online testing.

INDEX TERMS Mutual dependence, pattern matching, random sequence, sequence alignment, series similarity.

I. INTRODUCTION

Sequence analysis is a broad and important field of science and technology, where digital signal processing in information engineering is at the forefront. In this area, the sub-field of comparing random sequences, although developed over the past years [1], is recently gaining importance particularly quickly, creating a separate domain in cryptography [2]. Therefore, a special case of sequence analysis is a comparison of two sequences to answer the question, how are they similar to each other and what is the peculiarity of their mutual relationship. Due to the increasing demand for comparative analysis, the discipline has developed tremendously in recent years [3]–[6]. There are many different methods of assessing sequence alignment and similarity, the most intuitive of which still seems to be the correlation and mutual information between sequences [7], [8]. Both methods are widely used in telecommunications and cryptography to assess, e.g., randomness of a signal [7], [9], [10]. However, they are of limited use when simultaneous analysis of the temporal and statistical properties of data sequences is required. As cryptographic systems play an important part in various applications, e.g., ensuring secure data transmission, it is imperative that the provided sequences of random numbers are of appropriate quality, i.e. expected to be possibly close to true randomness. Therefore, sequences are subject to statistical testing, e.g. defined in NIST SP 800-22 [11] or AIS-31 [12] standards. However, statistical tests do not distinguish between truly random and high-quality pseudo-random sequences. As such, they cannot be used, e.g., to assess within a reasonable time whether the generator is under attack. A convenient solution for examining the determinism is the mechanism proposed in [13], based on repeatedly starting the generator with the same initial conditions. The described testing technique is called the restart mechanism and can help to check if the generator produces sequences as a result of deterministic or non-deterministic phenomena. On the collected data set, the chi-square test [14], standard deviation [15], or entropy estimation is applied [16]. However, the restart mechanism and following statistical tests require collecting many data sequences, and thus are difficult to apply in online sequence comparison and generator testing. Therefore, obtaining a low-data demanding yet efficient method that processes the currently obtained string against a single reference and returns easy-to-evaluate information about the relationship of sequences remains an open issue.

The solution to the considered problem may be sought in areas facing similar challenges. The problem of assessing

The associate editor coordinating the review of this manuscript and approving it for publication was Jun Shi.

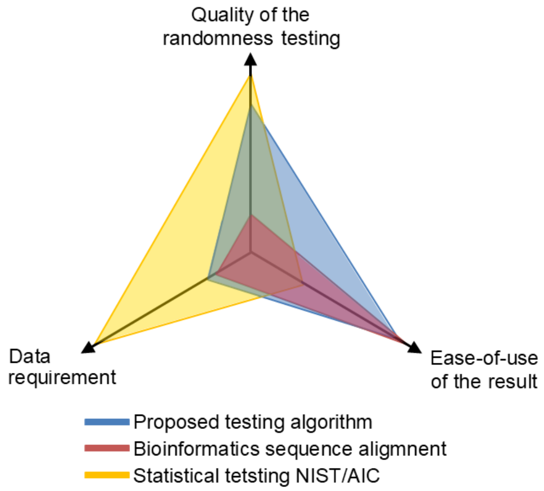

FIGURE 1. Characteristics of the proposed algorithm in comparison with available solutions.

how two sequences are similar to each other is also known in the analysis of natural language processing [17], [18]. The quantification of the similarity between texts is not unique and unambiguous, and largely depends on the relative importance attached to individual particles, letters, words, phonemes, and grammar, and even on the general context of its occurrence. More detailed descriptions of the methods can be found in the work [19]. A related issue is also raised in the science of molecular biology, where sequence comparison is required in comparing primary biological sequence information [20], such as protein, amino-acid, DNA, RNA, etc. Methods used in biological research are mainly based on word frequency, distances defined in Cartesian space by frequency vectors, and information contained in the frequency distribution [21]. Statistical algorithms involve, i.a., implicit Markov models, Bayesian methods for hypothesis testing, Kolmogorov's complexity and chaos theory [22]. What is more interesting in the context of considered problem is, however, a narrow group of sequence alignment methods. These methods are used in arranging molecular sequences to identify regions of similarity which may be a consequence of functional, structural, or evolutionary relationships between the sequences. The use of alignment comparison appears in numerous bioinformatics applications related to searching for a template in a database, where similarity is used to infer congruent structure or function [23].

Motivated by the desired functionality of random sequence testing and inspired by the above-mentioned algorithms, we propose an innovative algorithm for sequence comparative analysis. It combines the features of alignment, alignment-free, and information theory sequence comparison techniques, and is designed to compare random sequences produced by the random number generator - one of the most important elements required in cryptographic systems, and based on a small set of input data return a reliable, easy-to-evaluate result (Fig. 1).

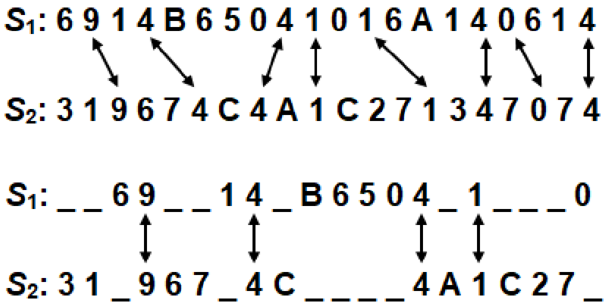
S_1: 6 9 1 4 B 6 5 0 4 1 0 1 6 A 1 4 0 6 1 4
S_2: 3 1 9 6 7 4 C 4 A 1 C 2 7 1 3 4 7 0 7 4
S_1: _ _ 6 9 _ _ 1 4 _ B 6 5 0 4 _ 1 _ _ _ 0
S_2: 3 1 _ 9 6 7 _ 4 C _ _ _ _ 4 A 1 C 2 7 _

FIGURE 2. An example of two random sequences with common elements (top) adjusted by the algorithm (bottom).

The algorithm proposed in this paper requires only two sequences of any length and tries to match one to the other by applying sequence breaks to matchless elements in the function of exponential cost. As a result, sequences derived from a non-deterministic phenomenon generate a high-cost alignment, while for a low-cost sequences the introduced interruptions allow inferring the nature of the dependence between the sequences. After applying the algorithm, the results are obtained in the form of two metrics: stretch ratio and stretch cost, which makes the proposed solution easy to use and interpret. Moreover, the algorithm returns the stretched sequences, providing readily available material for further match and mismatch statistical analysis.

The rest of the article is organized as follows. Section II describes the proposed alignment algorithm. Section III explains the experiment methodology and shows numerical results. Finally, Section IV gives the concluding remarks.

II. ALGORITHM DESCRIPTION

The algorithm proposed in the paper performs mutual matching of two random sequences by successive searching for common elements in both sequences and inserting gaps for matchless ones (Fig. 2). An important feature of the algorithm is the careful minimization of the introduced gaps. The algorithm starts with the collection of two random sequences S_1 and S_2. Then it successively compares the elements of both sequences. In the case of a match, i.e., $S_1[k] = S_2[k]$, it increments index k and simply goes to the next comparison. In the case of a mismatch, i.e., $S_1[k] \neq S_2[k]$, it introduces an auxiliary index $p = k$. Then it begins the search through the sequence S_1 by increasing index p, so as to find an element matching the current element of S_2, i.e., $S_1[p] = S_2[k]$, where $p > k$. Such an initial match generated in a straightforward way would require the recognition that $p - k$ elements from S_1 do not have their counterparts in S_2, so the latter should be filled with $g = p - k$ gaps. The above represents the worst possible case where gaps are inserted in series only into sequence S_2. Therefore, the algorithm checks whether any of the elements preceding $S_1[p]$, i.e., those falling within the range k to p, match any element following $S_2[k]$ in the same range of indexes. The further operation of the algorithm assumes transferring both fragments of sequences, that is $s_1 = S_1[k \dots p]$ and $s_2 = S_2[k \dots p]$, to a sub-algorithm searching for the optimal gapping in both subsequences (Fig. 3). Detailed block diagrams of the alignment algorithm

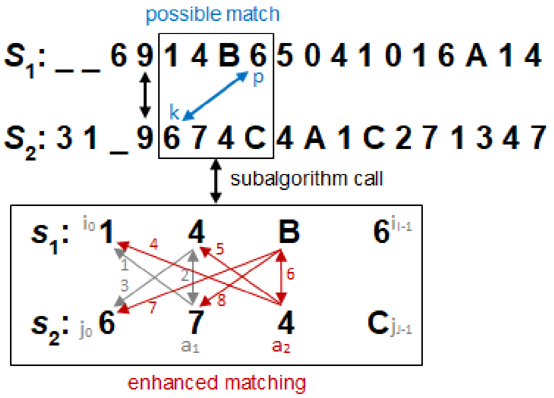

FIGURE 3. An example of the main algorithm operation with a call to enhanced matching subalgorithm.

and the sub-algorithm for optimal sequences gapping are presented in Figures 4 and 5.

At this point it is worth noting that the search of the pair $S_1[p] = S_2[k]$ is primarily aimed at limiting the length of the sequences passed to the sub-algorithm. There is no limit to increasing index p, and in a critical case, e.g., searching for matchless element $S_2[k]$, the rest of both sequences will be passed to the optimizing sub-function. In the paper, the same set of values of both sequences is assumed, thus the presence of unique elements is minimized. However, the range of the search can be arbitrarily limited.

Searching for a better match of elements than the first in s_2 with the last in s_1 requires creating an appropriate cost function. The defined function should favour interrupting both sequences evenly, rather than introducing long series of gaps in just one of them. Therefore, the proposed sequence gapping assumes an exponential increase in the cost for series of gaps, i.e., for G_i consecutive gaps the cost equals $2^{(G_i-1)}$. The total cost of gap insertion into both sequences is therefore equal to $2^{(G_1-1)} + 2^{(G_2-1)}$.

The sub-algorithm starts by reading both subsequences s_1, s_2 and determining their current lengths I and J respectively. Note that in the case of continually extending sequences S_1 and S_2 their final lengths do not have to be equal. Thus the lengths of last sub-sequences may differ.

Initially, the algorithm determines the maximum cost of matching, i.e., for $s_1[I-1] = s_2[0]$. Then, for a rolling index a pointing the current reference element, i.e., $s_2[a]$ or $s_1[a]$, it searches for a match from the opposing sequence. The found element, with the lowest possible index i or j preceding a, that is $s_1[i]$ or $s_2[j]$, respectively, is equivalent to the minimization of the gapping cost.

An example of operation of the sub-algorithm is shown in Figure (3). An initial match $S_1[p] = S_2[k]$ would require the insertion of three gaps into S_2 between elements 9 and 6 with the cost of 2^2. The algorithm performs the search through subsequences s_1 and s_2 for a lower-cost match by a rolling comparison, i.e., $s_1[0]$ with $s_2[1]$, $s_2[1]$ with $s_1[1]$ and $s_2[0]$ with $s_1[1]$, and further $s_1[0]$ with $s_2[2]$, $s_1[1]$ with $s_2[2]$ etc. The approach enables it to find an optimal match, i.e., $s_1[1] = s_2[2]$, requiring the insertion of only one gap into s_1 and two gaps into s_2, thus generating the cost of $2^0 + 2^1$. The determined numbers of the gaps, marked as G_1 and G_2, are returned to the main algorithm.

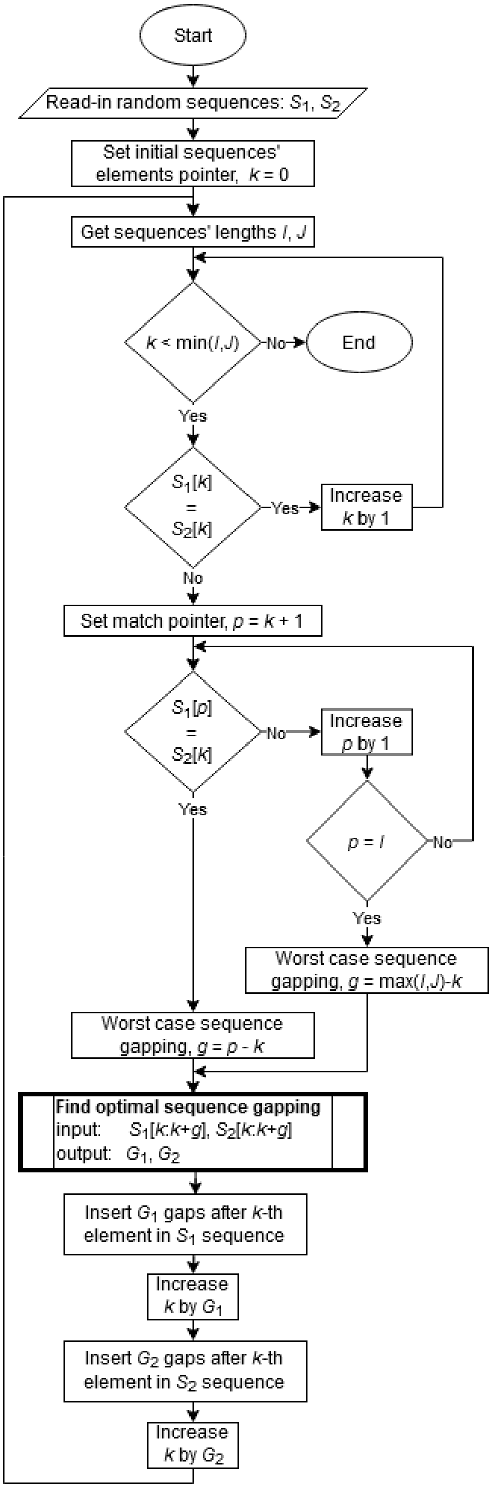
Start
Read-in random sequences: S1, S2
Set initial sequences' elements pointer, k = 0
Get sequences' lengths I, J
k < min(I,J)
No
End
Yes
S1[k] = S2[k]
Yes
Increase k by 1
No
Set match pointer, p = k + 1
S1[p] = S2[k]
No
Increase p by 1
p = I
No
Yes
Yes
Worst case sequence gapping, g = max(I,J)-k
Worst case sequence gapping, g = p - k
Find optimal sequence gapping
input: S1[k:k+g], S2[k:k+g]
output: G1, G2
Insert G1 gaps after k-th element in S1 sequence
Increase k by G1
Insert G2 gaps after k-th element in S2 sequence
Increase k by G2

FIGURE 4. Block diagram of the main algorithm for two-way sequence alignment.

At this point, it is worth noting that the sub-algorithm is complemented by conditions ensuring correct searching in the case of subsequences of different lengths. Moreover,

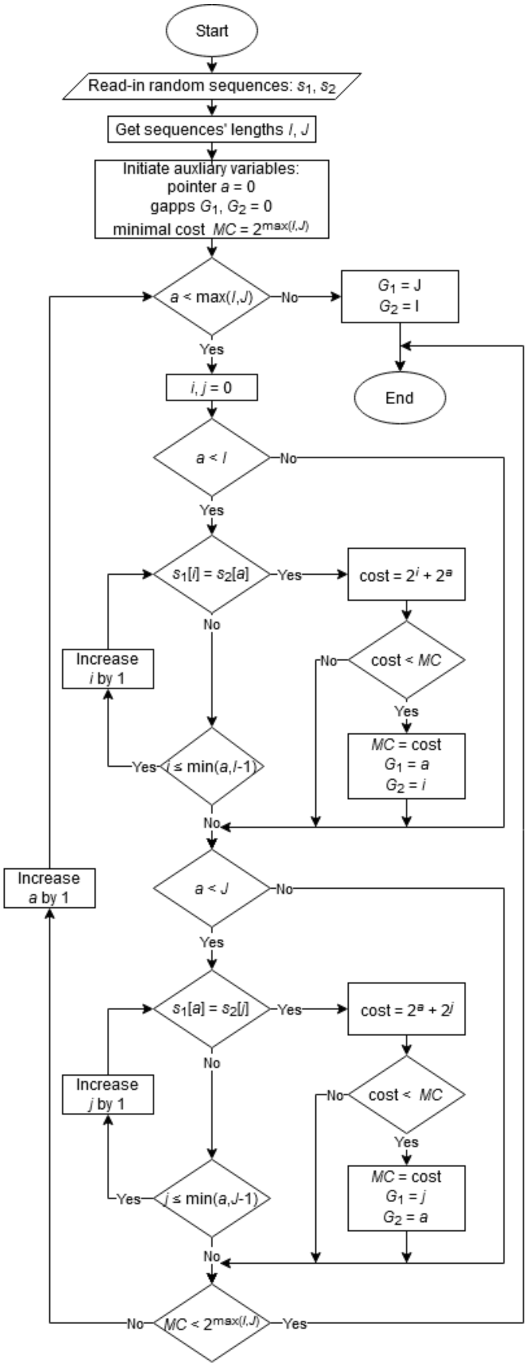

FIGURE 5. Sub-algorithm for finding optimal sequence gapping in the function of exponential cost.

in order to avoid the occurrence of large numbers, while determining the exponential cost function, for long series of gaps the operation may be performed on the exponents only. In this case, the algorithm will test the condition

$$i + a < 2(c - 1) \tag{1}$$

and alternatively

$$a + j < 2(c - 1) \tag{2}$$

where c is the exponent of the current minimum cost, i.e., $MC = 2^c$.

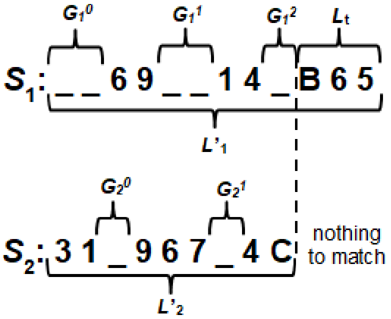

FIGURE 6. Sequences matched by the algorithm.

The main algorithm receives information on G_1, adds an adequate number of gaps after element $S_1[k]$ and updates the index to the position $k = k + G_1$. Then it receives the information on G_2 and inserts an adequate number of gaps after $S_2[k]$. This way, both sequences are stretched so that gaps in S_1 are inserted at the mismatch positions of S_2, and gaps in S_2 are introduced for mismatched elements in already stretched S_1.

The final goal of the algorithm is to obtain two pairs of metrics $S_i(a_i, b_i)$, describing each of the tested sequences:

- a_i is a stretch ratio determined as the ratio of the length of the gapped sequence L'_i to its initial length L_i. Importantly, in both lengths, only the number of elements to be matched is taken into account (Figure 5). As a result of the algorithm, the sequences increase in lengths dynamically. Consequently, in one of the sequences, a "tail" of length L_t may be created. L_t is the number of elements for which there were no more elements to match in the opposite sequence. Considering the above, the stretch ratio is redefined as

$$a_i = \left(L'_i - L_t\right) / \left(L_i - L_t\right) \tag{3}$$

- b_i is the cost of stretching the sequence, calculated as

$$b_i = \sum_j 2^{G_i^j - 1} / \left(L_i - L_t\right) \tag{4}$$

where j indexes successive series of gaps, and G_i^j is equal to the number of gaps in the j-th series.

Figure 6 and equations (5) to (8) show the process of computing the metrics for short sample sequences $S_1 = [6, 9, 1, 4, B, 6, 5]$ of the initial length L_1 and $S_2 = [3, 1, 9, 6, 7, 4, C]$ of the initial length L_2.

The metric for the first sequence $S_1(a_1, b_1)$ in the above example would be calculated as

$$a_1 = \left(L'_1 - L_t\right) / \left(L_1 - L_t\right) \tag{5}$$

$$b_1 = \frac{2^{G_1^0 - 1} + 2^{G_1^1 - 1} + 2^{G_1^2 - 1}}{L_1 - L_t} \tag{6}$$

whereas for the second sequence $S_2(a_2, b_2)$

$$a_2 = L'_2 / L_2 \tag{7}$$

$$b_1 = \frac{2^{G_2^0 - 1} + 2^{G_2^1 - 1}}{L_2} \tag{8}$$

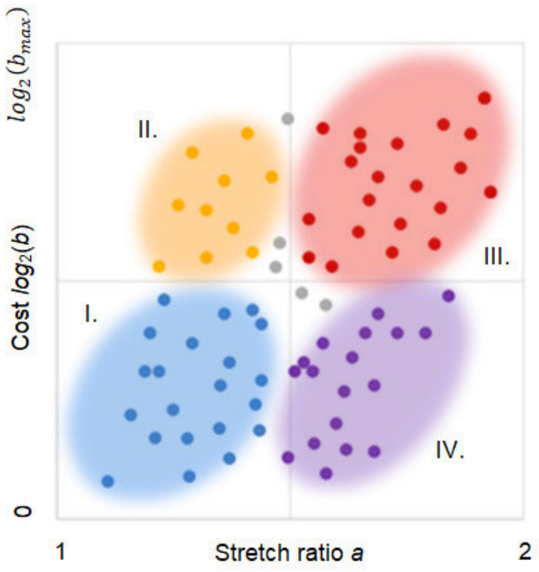

FIGURE 7. The meaning groups of the $S(a, b)$ metric in the plane of stretch and cost.

Both metrics have their strict limits. The maximum extension of one sequence may be the number of elements of the other. Therefore, $S_1(a_1, b_1)$ metrics should be in the range

$$a_1 \in \langle 1, \frac{L_1 + L_2 - L_t}{L_2} \rangle \tag{9}$$

$$b_1 \in \langle 0, \frac{2^{L_2 - 1}}{L_1 - L_t} \rangle \tag{10}$$

An analogous relationship with the length of S_1 takes place in the case of $S_2(a_2, b_2)$.

Considering the operation of the alignment algorithm, i.e., preferring interrupting both sequences evenly, the obtained $S_i(a_i, b_i)$ metrics are expected to be similar. Therefore, it is convenient to interpret a collective metric $S(a, b)$, where

$$a = \frac{L'_1 + L'_2 - L_t}{L_1 + L_2 - L_t} \tag{11}$$

$$b = \frac{\sum_i \sum_j 2^{G_i^j - 1}}{L_1 + L_2 - L_t} \tag{12}$$

In the above case, the stretch ratio $a \in \langle 1, 2 \rangle$ is a measure informing about the mutual similarity of both random sequences, whereas the cost of stretch $b \in \langle 0, \left(2^{L_1 - 1} + 2^{L_2 - 1}\right) / (L_1 + L2) \rangle$ informs about the average cost of processing each element, where a match costs 0. The described ranges reach given maximum values when all elements of the sequences are aligned and no tail is formed, i.e., $L_t = 0$. We can interpret the collective metric $S(a, b)$ in an a, b plane, as shown in Figure 7.

The metric can be classified into one of four general informative groups:

I low stretch ratio indicates high similarity of sequences. Low cost is indicative of scattered and short interruptions, therefore rare mismatches.

II low stretch ratio again means high similarity, while high cost informs about the insertion of gaps in series, i.e. cyclical convergences and divergences of both sequences.

III high stretch ratio and high cost are the result of introducing long series of gaps interleaved with sparse matches, thus informing about the negligible similarity of both sequences.

IV high stretch ratio while maintaining low cost indicates frequent insertion of single gaps, i.e., scattered short matches and mismatches. The result is a premise for inference about the statistical similarity of the sequences.

III. EXPERIMENT

To provide proof of the concept, test data were prepared. Each test sequence has the length of 5000 elements, which take values from 0 to 15, and it corresponds to a four-bit representation. Section III-A presents the comparisons of pseudorandom sequences and section III-B shows the comparisons of sequences from hardware random number generators based on a Fibonacci Ring Oscillator (FIRO) [13].

A. PSEUDORANDOM SEQUENCES

In the first step of the test, pseudorandom number sequences were generated with specific distributions. Four distributions were chosen: Gaussian, Uniform, Rayleigh, and Poisson. Sequences with the Gaussian distribution were generated with the mean value equal to two, four, eight, and fourteen. In sequences with the Uniform distribution, values from zero to fifteen occur with the same probability. For the Poisson distribution, the lambda parameter equal to two was chosen and for the Rayleigh distribution, the sigma parameter equal to two was set up. Compliance with the prepared data against intended results was checked by plotting and verifying histograms of the sequences. Additionally, to achieve certain statistic and time properties, some manual modifications of the sequences were made, i.e., in a sequence with the Gaussian distribution with the mean value of two and sigma of one, non-matching series of elements were inserted, and the extended length was truncated back to 5000. The used non-matching value was seven, as it did not occur in the original file. Insertions of lengths 100, 500, 1000, 2000, 3000, and 4000 were made. In total, 40 different files were prepared to check the properties of the metrics. Then the cost metric and the stretch ratio were calculated for all possible combinations of the files. Figure 8 presents all 1600 results in the stretch-cost plane. Data presented in Table 1 show how the inserted non-matching elements influence the cost and the stretch ratio. The dots highlighted in red show the data presented in Table 1.

The cost corresponds to the size of the mismatched series indicated by continuous gaps, while increasing the gap size increases the stretch ratio, which is inversely proportional to the percentage of matches. For example, when all values match, the stretch ratio is equal to one, and when there are no matches, it is equal to two.

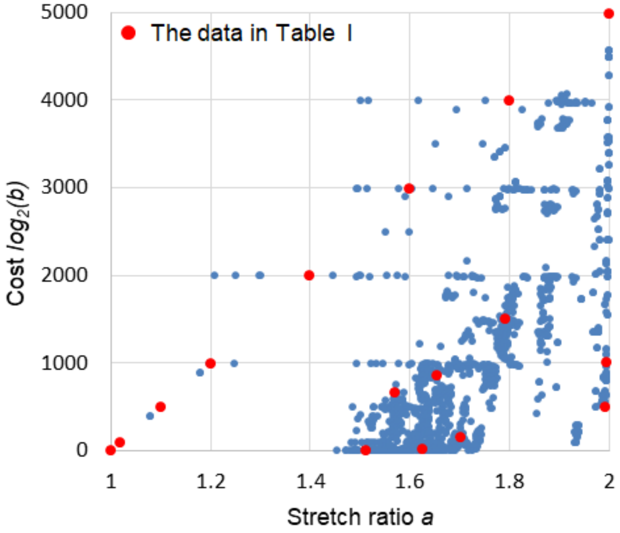

FIGURE 8. The pseudorandom sequences comparison results in the stretch-cost plane.

TABLE 1. Experiment results of pseudorandom sequences comparison.

Sequence in comparison with Gauss $N(2,1)$		Results	
Distribution	Comments	Stretch ratio a	Cost $log_2(b)$
Gauss $N(2,1)$	0 INS	1	0
Gauss $N(2,1)$	100 INS	1.02	86.71
Gauss $N(2,1)$	500 INS	1.10	486.71
Gauss $N(2,1)$	1000 INS	1.20	986.71
Gauss $N(2,1)$	2000 INS	1.40	1986.71
Gauss $N(2,1)$	3000 INS	1.60	2986.71
Gauss $N(2,1)$	4000 INS	1.80	3986.71
Rayleigh $R(2)$	-	1.51	3
Poisson $P(2)$	-	1.57	654.71
Uniform $U(0,15)$	-	1.62	12.73
Rayleigh $R(3)$	-	1.65	853.71
Gauss $N(4,1)$	-	1.70	144.71
Uniform $U(0,15)$	-	1.79	1504.71
Gauss $N(8,1)$	-	1.99	491.71
Gauss $N(8,1)$	-	1.99	1009.71
Gauss $N(14,1)$	-	2	4985.76

ΔF^ises; INS - in the sequence of certain distribution some non-matching insertions were made.

A comparison of different sequences with Gaussian distributions shows that when the mean value difference is small, e.g., for a mean of two and mean of four, the stretch ratio remains low, but when the difference grows, and thus fewer values of both distributions overlap, then the stretch ratio also increases, reaching a maximum for the most distant pair, i.e., with the mean of two and the mean of fourteen. The effect is visible for different distributions as well, e.g., the sequence with Gaussian distribution and the mean of two is similar to data with Rayleigh distribution and sigma of two, therefore the stretch ratio for this pair is relatively low.

Data points presented in Figure 8 confirm the interpretation of the groups presented in Figure 7. The stretch ratio shows the coincidence of both sequences, while the cost describes the matching pattern. Low cost indicates the insertion of short gaps series, and high cost informs about long breaks. Figure 8 shows that for a stretch ratio close to maximum, the cost varies from 500 to 5000.

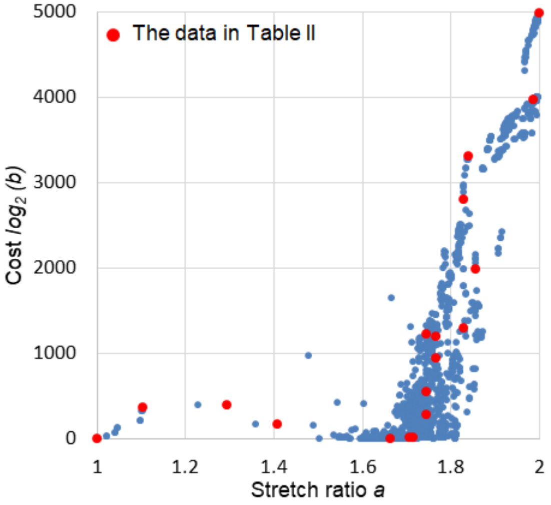

FIGURE 9. The FIRO random generator comparison results in the stretch-cost plane.

TABLE 2. Experiment results of true random sequences comparison.

Sequence 1		Sequence 2		Results	
Source	Id^i	Source	Id^i	Stretch ratio a	Cost $log_2(b)$
FPGA	d3_r1	FPGA	d3_r1	1	0
FPGA	d3_r1	FPGA	d3_r3	1.29	395
FPGA	d3_r1	FPGA	d17_r3	1.70	7.39
FPGA	d3_r1	FPGA	d99_r3	1.74	280
FPGA	d3_r1	ISE	d4_c3	1.77	940.85
FPGA	d3_r1	ISE	d4_c1	1.83	2793.85
FPGA	d3_r1	FPGA	d25_r1	1.99	3960
FPGA	d25_r1	FPGA	d25_r4	1.10	359
FPGA	d5_r4	FPGA	d5_r1	1.40	159
ISE	d1_c3	FPGA	d49_r4	1.66	1.58
ISE	d1_c3	FPGA	d3_r3	1.72	3.58
ISE	d1_c3	FPGA	d17_r1	1.75	1215.85
ISE	d1_c3	ISE	d3_c1	1.75	544.71
ISE	d1_c3	FPGA	d99_r4	1.77	1196.85
ISE	d1_c3	FPGA	d5_r1	1.83	1287.85
ISE	d1_c3	FPGA	d25_r1	1.84	3300.85
ISE	d1_c3	ISE	d4_c1	1.86	1980.71
ISE	d1_c3	ISE	d2_c3	1.99	4983.71

[i] Meaning of phrases; according to files d-delay, r-restart, and c-clock.

On the other hand, for the cost of 2000, the stretch ratio varies from 1.2 to 2. Thus, the obtained results fit into the predicted state-space in the a, b plane.

B. TRUE RANDOM SEQUENCES

The second part of the experiment was focused on testing random sequences obtained from hardware random number generators. A major part of the sequences comes from a hardware generator implemented in an FPGA – Spartan 6 XC6LX16. The data set is supplemented with pseudo-random sequences from the Xilinx ISE Design Suite circuit simulator. Again, 40 different sequences were prepared, and all possible combinations of the pairs were examined. All results are presented in Figure 9 and the data in Table 2 are highlighted in red.

Table 2 shows that in the control case, i.e., self-comparison of a sequence, the match cost is 0 and the stretch ratio is 1. However, for different sequences obtained from the same generator, the stretch ratio remains low with the match cost lower than 500. Careful analysis of the results clearly indicates sequences similar to each other, i.e., those having a

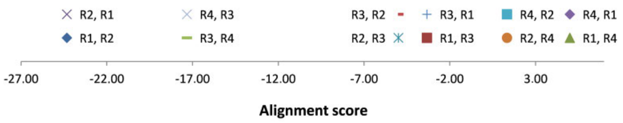

FIGURE 10. The results of the Needleman-Wunsch alignment algorithm.

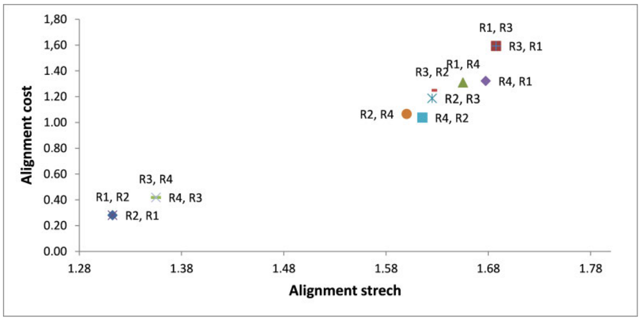

FIGURE 11. The results of the proposed alignment algorithm.

match cost lower than 100 and a stretch ratio close to 1.7. By analogy, totally different sequences reach the stretch ratio close to 2 and the cost higher than 3000. When to different pairs of sequences, the same stretch ratio is assigned, e.g., close to 1.75, it is possible to distinguish the nature of their mutual convergence using a second metric. In such a case, the match cost varies from 0 to 1500. Using two metrics simultaneously provides a better overview of the similar type of both sequences.

The experimental results described above cover different types of dependencies, and once more confirm the expected performance of the proposed algorithm.

The results provided in Tables 1 and 2 demonstrate the wide application of the algorithm. The proposed method can be an effective tool for direct sequence comparison, and thus randomness assessment. Moreover, it can be used as a selector of structures and constructions which provide the highest match cost and stretch ratio, to ensure the high quality of generated random sequences. The metric can also be used as a general-purpose verification tool to compare the simulation results with the gathered real data.

C. METHODS COMPARISON

Although the proposed method is derived from bioinformatics, it significantly extends the matching methodology. To show the fundamental difference and its influence on the alignment result, we compare the operation of the algorithm with the well-known Needleman-Wunsch (NW) technique [24]. For comparison purposes, four 128-bit random sequences, i.e., R1-R4, were generated. The test data was interpreted as strings of 16 characters mapped to the form required by the given algorithm. In order to ensure the appropriate randomness, generators based on ring oscillators, implemented in Intel MAX10 and Xilinx Spartan 6 devices, were used. The numerical results of the alignment are shown in Figures 10 and 11.

Figure 10 shows the match score from the NW algorithm. To systematize the interpretation of the alignment in the plot, the obtained score was negated, thus the minimum value

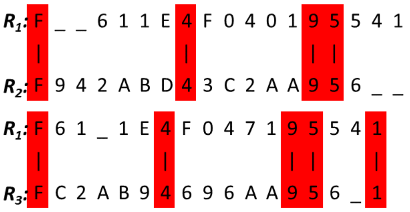
R1: F _ _ 6 1 1 E 4 F 0 4 0 1 9 5 5 4 1
R2: F 9 4 2 A B D 4 3 C 2 A A 9 5 6 _ _
R1: F 6 1 _ 1 E 4 F 0 4 7 1 9 5 5 4 1
R3: F C 2 A B 9 4 6 9 6 A A 9 5 6 _ 1

FIGURE 12. Alignment of the (R1,R2) and (R1,R3) pairs using the Needleman-Wunsch algorithm.

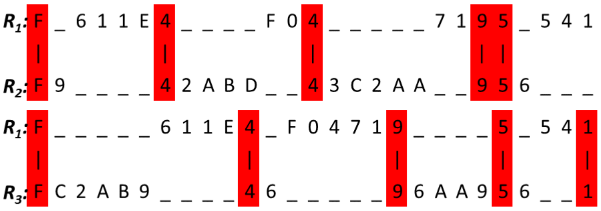
R1: F _ 6 1 1 E 4 _ _ _ _ F 0 4 _ _ _ _ _ 7 1 9 5 _ 5 4 1
R2: F 9 _ _ _ _ 4 2 A B D _ _ 4 3 C 2 A A _ _ 9 5 6 _ _ _
R1: F _ _ _ _ _ 6 1 1 E 4 _ F 0 4 7 1 9 _ _ _ _ 5 _ 5 4 1
R3: F C 2 A B 9 _ _ _ _ 4 6 _ _ _ _ _ 9 6 A A 9 5 6 _ _ 1

FIGURE 13. Alignment of the (R1,R2) and (R1,R3) pairs using the proposed algorithm.

indicates the greatest similarity between the sequences, and the maximum indicates the most differentiated pair. In the discussed case, we can see that the NW algorithm is symmetric, indicating (R1,R2) as the most similar and (R1,R4) as the most different.

Figure 11 shows the results for the proposed method. The obtained metrics exhibit a linear tendency, reproducing the order and symmetry of pairs (R1,R2), (R3, R4), and (R1,R3) shown by the NW algorithm. However, there is an important difference for the sequences (R2,R4) and (R1,R4). In these cases, a difference in the stretch is due to the effect of producing the "tail", i.e., end mismatches. Moreover, pairs (R1,R4) and (R2,R4) were rearranged in comparison with the ordering of the NW results. Figures 12 and 13 explain the source of the differences in detail.

In the considered case, the proposed algorithm, unlike NW, recognizes the pair (R1,R4) as better suited than (R1,R3). This is due to finding one more match in (R1,R4) than through NW. However, for the comparison of (R1,R4) and (R1,R3) – both with 5 matches, the most important is the exponential cost of the alignment instead of linear. In the case of the pair (R1,R3) it is required to insert longer gaps series than in the pair (R1,R4). Whereas in the NW algorithm this difference is insignificant. This case shows that the proposed algorithm is as effective as NW in the sequence alignment. However, by determining a more distinguishable cost function, it provides additional information allowing not only a binary evaluation but also an interpretation of the sequence alignment properties.

IV. CONCLUSION

The algorithm described in the paper provides the results of a comparative analysis in the form of the $S(a, b)$ metric, which primarily ensures easy interpretation. The returned metric indicates the degree of mutual similarity of the sequences and allows one to infer the nature of their dependence. The expected behavior of the algorithm was confirmed both by simulation, using strings with known properties, and experimentally with the use of hardware random number generators with a known structure.

An important feature of the algorithm is its simple implementation, based on iterations, increments, and comparisons. This provides easy hardware realization as a build-in block in FPGAs and ASICs, whereas the use of a single reference enables the currently obtained sequence to be examined even in soft real-time systems. Therefore, the algorithm can be effectively used to follow the current operating state of the generator, and successfully used at the design stage as well, e.g., in the analysis of the similarity of sub-generators in more complex structures.

REFERENCES

[1] L. Blum, M. Blum, and M. Shub, "Comparison of two pseudo-random number generators," in *Advances in Cryptology*. Santa Barbara, CA, USA: Plenum, 1982, pp. 62–78.

[2] J. Wisniewska, "Comparing quality of pseudo- and true-random numbers obtained from different sources," in *Proc. 25th Sci. Workshops Polish Soc. Comput. Simul. (PTSK)*, Ryn, Poland, 2018, pp. 1–11.

[3] F. Yutao and S. Guiping, "Application of wavelet analysis in removing deviation and correlation from true random sequences," in *Proc. IEEE 5th Int. Conf. Softw. Eng. Service Sci.*, Jun. 2014, pp. 545–548.

[4] S. Popereshniak, "The testing of pseudorandom sequence of small length as a component of the Internet of Things security," in *Proc. IEEE Int. Conf. Adv. Trends Inf. Theory (ATIT)*, Dec. 2019, pp. 277–281.

[5] I. Stankovic, C. Ioana, and M. Dakovic, "Sequence comparison in reconstruction and targeting in underwater sonar imaging," in *Proc. OCEANS Marseille*, Jun. 2019, pp. 1–10.

[6] K. Li, J. Zhang, P. Li, A. Wang, and Y. Wang, "Parallel implementation of the non-overlapping template matching test using CUDA," *China Commun.*, vol. 17, no. 8, pp. 234–241, Aug. 2020.

[7] D. M. Horan and R. A. Guinee, "Correlation analysis of random number sequences based on pseudo random binary sequence generation," in *Proc. IEEE Inf. Theory Workshop*, Sep. 2005, p. 5.

[8] N. Jain and C. A. Murthy, "A new estimate of mutual information based measure of dependence between two variables: Properties and fast implementation," *Int. J. Mach. Learn. Cybern.*, vol. 7, no. 5, pp. 857–875, Oct. 2016.

[9] A. Compagner, "The hierarchy of correlations in random binary sequences," *J. Stat. Phys.*, vol. 63, pp. 883–896, Jun. 1991.

[10] R. M. Gray and P. C. Shields, "The maximum mutual information between two random processes," *Inf. Control*, vol. 33, no. 4, pp. 273–280, Apr. 1977.

[11] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," NIST Special Publication, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. 800-22 Revision 1a, 2010.

[12] W. Schindler and W. Killmann, "Evaluation criteria for true (physical) random number generators used in cryptographic applications," in *Cryptographic Hardware and Embedded Systems—CHES* (Lecture Notes in Computer Science), vol. 2523, B. Kaliski, C. Koc and C. Paar, Eds. Berlin, Germany: Springer, 2003, pp. 431–449.

[13] M. Dichtl and J. D. Golic, "High-speed true random number generation with logic gates only," in *Proc. Cryptograph. Hardw. Embedded Syst. (CHES)* Vienna, Austria, Sep. 2007, pp. 45–62.

[14] M. Jessa and L. Matuszewski, "Enhancing the randomness of a combined true random number generator based on the ring oscillator sampling method," in *Proc. Int. Conf. Reconfigurable Comput. FPGAs*, Nov. 2011, pp. 274–279.

[15] X. Xu and Y. Wang, "High speed true random number generator based on FPGA," in *Proc. Int. Conf. Inf. Syst. Eng. (ICISE)*, Apr. 2016, pp. 18–21.

[16] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle, "Recommendation for the entropy sources used for random bit generation," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NIST Special Publication 800-90B, 2018.

[17] D. B. Searls, "Reading the book of life," *Bioinformatics*, vol. 17, no. 7, pp. 579–580, 2001.

[18] J. P. Pestian, L. Deleger, G. K. Savova, J. W. Dexheimer, and I. Solti, "Natural language processing—The basics," in *Pediatric Biomedical Informatics*. (Translational Bioinformatics), vol. 2, J. Hutton, Ed. New York, NY, USA: Springer, 2012.

[19] S. Vinga and J. Almeida, "Alignment-free sequence comparison—A review," *Bioinformatics*, vol. 19, no. 4, pp. 513—523, 2003.

[20] R. Fuchs, "From sequence to biology: The impact on bioinformatics," *Bioinformatics*, vol. 18, no. 4, pp. 505–506, 2002.

[21] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge, U.K.: Cambridge Univ. Press, 1997.

[22] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang, "An information-based sequence distance and its application to whole mitochondrial genome phylogeny," *Bioinformatics*, vol. 17, no. 2, pp. 149–154, 2001.

[23] W. R. Pearson, *Protein Sequence Comparison and Proteinevolution. Tutorial-ISMB2000*. Charlottesville, VA, USA: Univ. Virginia, 2000.

[24] S. Angizi, J. Sun, W. Zhang, and D. Fan, "PIM-Aligner: A processing-in-MRAM platform for biological sequence alignment," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 1265–1270.

JAKUB NIKONOWICZ was born in Poland, in 1990. He received the M.Sc. degree in electronics and telecommunication and the Ph.D. degree (Hons.) in telecommunication from the Poznań University of Technology, Poznań, Poland, in 2014 and 2019, respectively. He has authored or coauthored 12 scientific publications in refereed journals and proceedings of international conferences. His current research interests include statistical signal processing for blind signal detection and random number generation for reliable node authorization in wireless sensor networks.

ŁUKASZ MATUSZEWSKI graduated from the Faculty of Electronics and Telecommunications, Poznań University of Technology, in 2010. He defended the Ph.D. thesis entitled "The Use of Reprogrammable Circuits to Generate Random Sequences," in 2019. He was employed with the Department of Telecommunications Systems and Optoelectronics, Faculty of Electronics and Telecommunications, Poznań University of Technology, in 2011. Since 2020, he has been an Assistant Professor with the Faculty of Computer Science and Telecommunications, Institute of Multimedia Telecommunications, Poznań University of Technology. He was a project manager for grants for young scientists, as well as the contractor in projects for a Polish telecommunications operator. He is the author or coauthor of 25 scientific publications in peer-reviewed journals and materials from national and international conferences. His research interests include designing devices with the use of reprogrammable circuits, in particular cryptographic circuits, random number generators, and synchronization circuits.

PAWEL KUBCZAK graduated from the Faculty of Electronics and Telecommunications, Poznań University of Technology, in 2013. He continues his studies at the Faculty of Electronics and Telecommunications, Poznań University of Technology. His research interests include digital random number generators, measuring the time period with picosecond accuracy, and programmable digital circuits.