An Efficient and Rapid Method for Detection of Mutations in Deoxyribonucleic Acid - Sequences

Wajih Rhalem[1], Jamal El Mhamdi[2], Mourad Raji[3]
E2SN-Laboratory
ENSET Mohammed V University, Rabat, Morocco

Ahmed Hammouch[4]
Direction of Scientific Research and Innovation
Ministry of Higher Education, Rabat, Morocco

Aqili Nabil[5]
LRGE Laboratory, ENSET Mohammed V University
Rabat, Morocco

Nassim Kharmoum[6]
Department of Computer Science
Intelligent Processing Systems and Security Team
Faculty of Sciences, Mohammed V University
Rabat, Morocco

Hassan Ghazal[7]
National Center for Scientific and Technical Research
Ministry of Higher Education Rabat, Morocco

***Abstract*—The comparison of genomic sequences plays a key role in determining the structural and functional relationships between genes. This comparison is carried out by identifying the similarities, differences and mutations between genomic sequences. This makes it possible to study and analyze the genetic and the evolutionary relationships between organisms. Alignment algorithms have been in the spotlight for the last few decades, due to a vast genomic data explosion. They have attracted a great deal of interest from many researchers who focus on the development of practical solutions to ensure effective alignments with an optimal response time. In this paper, a novel algorithm based on Discrete To Continuous "DTC" approach has been developed. The proposed methodology was compared against other existing methods, which are largely based on the concept of string matching. Experimental results show that the DTC algorithm delivers supremely efficient alignment with a reduced response time.**

Keywords—Alignment algorithms; genomic sequences; dynamic polynomial interpolation; mutations

I. Introduction

Bioinformatics is the intersection of biology and informatics because it is a field that covers life sciences disciplines such as genomics, proteomics and biology through computer methods. The main mission of this research area is to analyze and interpret deoxyribonucleic acid (DNA) sequences in central databases, accessible worldwide, to enable scientists to present and search biological information.

The DNA sequence is an ordered collection of alphabets of the four nucleotides A, C, G and T containing the information necessary for the survival and reproduction of living beings. Analyzing this sequence is then important and useful for both research on the life of organisms and for biomedical engineering.

Comparison of DNA sequences is done through softwares based on alignment algorithms that give results in the form of score and percentages of similarities and identities, and whose dynamic programming plays a considerable role. Dynamic programming relies on a relationship between the optimal solution of the problem and that of a finite number of sub-problems. Concretely, this means that it would be possible to deduce the optimal solution of a problem from an optimal solution of a sub-problem.

With regard to sequence alignment, three types of alignment of the DNA sequences can be distinguished:

1) Global alignment: used when the sequences are about the same length because the alignment is done on all their lengths. This type of alignment was first proposed by Needleman and Wunsch [1].

2) Semi-global alignment: used in the case where one sequence is shorter than the other or when one looks for overlaps at the ends without counting the penalties of the gaps.

3) Local Alignment: that searches for the two most conserved sub-regions between two sequences and only these two regions will be aligned. Smith and Waterman algorithm [2] is the most used in this matter.

In this study, a new DNA sequence alignment algorithm Discrete-To-Continuous (DTC), will be presented to ensure the three types of alignment: Global, Semi Global and Local. DTC relies on dynamic programming based on polynomial interpolation of data. This approach was originally applied in shape recognition and chirality measurement [3]. Subsequently, DTC has been adapted to tackle other areas of application, namely: the alignment of time-shifted signals [4], correction of the DNA-electropherogram errors resulting from capillary electrophoresis sequencing experiments [5], online signature matching [6], speech recognition [7], algorithmic geometry [8] and fingerprint matching [9].

Unlike string matching algorithms, which try to find a point-to-point correspondence of the chains, the DTC approach solves this problem in its entirety by superimposing the discrete representation of the test points on the continuous representation of the reference points. In section IV, the operating principle of the algorithm will be presented in detail.

In order to ensure the performance of this approach, the programming and adaptation of DTC in the DNA sequence alignment domain were carried out. For this purpose, a comparative study was carried out in terms of accuracy, temporal complexity and response time with other algorithms that were the subject of a benchmarking study on a DNA sequence. The sequence studied in this work is JN222368 for Genbank belonging to the marine sponge.

For an efficient comparison, the test environment and conditions were unified by downloading a partial Genbank database containing 7682 sequences of different sizes, including the sequence in question JN222368. Subsequently, the DTC algorithm was implemented as well as the other reference algorithms in the Java programming language. The machine used was a 2.40 GHz Intel Core i7 processor with 8 GB of RAM. Experimental results show that the DTC algoritm delivers supremely efficient alignment with a reduced response time, including in the detection of mutations and gaps.

This work is organized as follows. Section I is a general introduction of the problem. Section II is dedicated to presenting the studies and works carried out during the last five years in this area of competence. Section III gives an overview of the different string matching algorithms. Section IV presents, in a detailed and in-depth way, the operating principle of the DTC approach. The results of comparing DTC algorithm with the other approaches are presented and discussed in Section V.

II. Related Works

Given the importance of string matching algorithms, in determining the functional and structural relationships of the biological sequence, several studies and works have been carried out. This section is dedicated to presenting the studies and works carried out during the last five years in this area of competence:

In 2015, a research team made up of professors: Nadia B.N, Lecroq T and Elloumi M, conducted a study [10] presenting an algorithm which extends the variants of Boyer-Moore's exact string matching algorithm. The goal of this work is to solve the problem of exact pattern matching in a set of similar DNA sequences, in which only the pattern can be preprocessed.

In another work carried out in 2015 [11], new methods for matching key motifs in secondary RNA structures, based on the notion of structural chains, were proposed. In this approach, new correspondence algorithms to solve the problem of structural matching problem were used. This solution also made it possible to respond to various combinatorial requests encountered during the pairing of secondary RNA structures.

In 2017 a comparative study [12] was performed on exact string matching algorithms in the field of DNA sequence analysis by Iji and Mahalakshmi. This work was essentially based on the response time, the alignment accuracy of the DNA sequences and the temporal complexity of the algorithms in question. The results revealed that the Boyer-Moore algorithm provides the highest accuracy while the Reverse Colussi algorithm provides the shortest run time.

In another study carried out in 2019 by [13], a new solution was used, based on massive multithreaded exploitation with a focus on the latest Intel architectures based on Advanced Vector Extensions 512 (AVX-512). The goal is to address the limited acceptance of the Smith-Waterman algorithm by the computational requirements of large protein databases often used for local sequence alignment.

Recently in 2019, a new treatment method [14] based on the comparison of sequences without using explicit pair pairing, was proposed by S. Kouchaki, A. Tapinos and D. L. Robertson.This approach provides a viable solution to the functions of the textual representation of sequences data.

One of the most recent studies in this area was done in 2020. In this study [15], an algorithm called Maximal Average Shift (MAS) was presented. Its operating principle consists in finding a pattern scan order which maximizes the average length of the offset. In this work, two MAS extensions were also presented: the first optimizes the MAS scanning speed, by means of the result of the analysis in the previous window, while the second optimizes its processing time by deploying q-grams. The results of this study revealed that these methods have better average scanning speed performance than previous chain matching algorithms for DNA sequences.

String Matching Algorithms

String matching algorithms play a key role in analyzing of biological sequences and they are divided into two categories:

1) The "exact string matching", whose algorithms are below, used to find the exact substring match; 2) The approximate match, which attempts to approximately find strings that correspond to a given pattern. The following algorithms: Rabin Karp [16] and [17], Brute Force [18] and Fuzzy string searching [19], are often used in this area of matching. In this section we will give a brief overview of string matching algorithms, focusing on their spatial and temporal complexities. Then a comparative study on said algorithms will be described.

A. Description of the String Matching Algorithms

1) Smith-Waterman algorithm [2]: This algorithm was invented by Temple F. Smith and Michael S. Waterman in 1981. It is often used in DNA sequence alignment, especially for gene prediction, phylogeny or function prediction. Its operating principle is to give an alignment corresponding to the best matching score between the nucleotides of the subject sequences. It relies on dynamic programming using similarity matrices or substitution matrices. Alignment is accomplished by inserting "gaps" or “INDELs” into the reference sequence or subject sequence in order to increase the number of matching characters between the two sequences. The preprocessing phase requires temporal ($m+ \sigma$) and spatial (σ)

complexities. The search phase of the algorithm requires a quadratic time complexity.

2) Needleman–Wunsch algorithm [1]: This algorithm is often used in the maximum global alignment of two character chains, especially protein or DNA sequences. The algorithm looks for the maximum score alignment. This was the first application of dynamic programming for the comparison of biological sequences. The processing time to search for a pattern in a given text is O(mn).

3) Boyer-Moore algorithm [20]: Boyer-Moore is considered one of the most commonly used string matching algorithms in everyday applications. The operating principle of this approach is based on the analysis of the characters of the text from right to left starting with the rightmost. If a complete match is detected, it deploys two precomputed functions to shift the window to the right, known by the matching shift and occurrence shift. The temporal complexity of Boyer Moore is of order O(mn).

4) Turbo-Boyer-Moore algorithm [21]: The Turbo-BM algorithm is a variant of the Boyer-Moore algorithm. Unlike the original Boyor More, this modified version does not require additional pretreatment and occupies only one constant additional space. It consists of recalling the text factor that corresponds to a suffix of the model during the last attempt, and this only in the case of a correct suffix offset. The peculiarity of this improvement is that it is possible to perform a turbo shift by neglecting said text factor. The temporal complexity of this algorithm is O (m.n).

5) Tuned Boyer-Moore Algorithm [22]: The Tuned Boyer-Moore is another variant of Boyer-Moore algorithm, intended to increase the speed of treatment. The principle of this approach is to optimize the matching verification phase between the character of the pattern and the character of the window. To avoid redoing this verification, which is very expensive in terms of response time, this method takes several shifts before performing a real characters comparison; the order of the comparisons between the characters of a pattern and text during each attempt not posing any more constraints. The temporal complexity of this algorithm is also of order O (mn).

6) Brute force algorithm [18]: The Brute force matching string algorithm is a classic alignment model, which does not require preprocessing. This approach attempts to verify, at all positions of the text, the position of occurrence of the pattern. The extracted patterns are compared one by one. The search window is moved exactly one position from right to left. The search can begin in any order (from left to right / from right to left). The temporal complexity of the search phase is equal to O (mn) and to a minimum comparison of 2 expected characters.

7) Deterministic Finite Automaton algorithm [23]: This algorithm consists of searching for a given sequence through the use of a finite state automaton. Each character in the model has a state, and each match sends the automaton to a new state. After matching all the characters in the pattern, the automaton switches to the approval state. In this case, the automaton will return to a suitable state depending on the primary state and the entered character. This algorithm has a temporal complexity of order O (n) since each character is examined once. This technique is very efficient because it examines each character of the text exactly once and displays all valid time shifts.

8) Karp-Rabin algorithm [17]: The Rabin-Karp algorithm calculates a numeric value (hash) for the pattern p and for each substring of m characters from text. Then, it confronts numerical values instead of confronting the real symbols. At the moment when a match is detected, the pattern is compared to the substring by a naive approach. If not, it goes to the next substring of the sequence to compare with p. The hash method deployed in this algorithm provides a simple process by avoiding a quadratic number of character comparisons in most practical situations. The time complexity of the algorithm is O(m+n).

9) Knuth Morris-Pratt algorithm [24]: This algorithm was developed by Morris and Pratt as the first linear time-match algorithm based on the analysis of the naive algorithm. The Knuth-Morris-Pratt algorithm preserves the information that the naive approach has consumed during the text analysis period. This approach avoids the exhaustion of the information through a temporal complexity of order ($m + n$). The use of this algorithm is effective because it minimizes the total number of comparisons of the pattern with the input string.

10)Reverse Colussi algorithm [25]: The Reverse Colussi string matching algorithm is another Boyer-Moore derivative. This algorithm consists of partitioning all the positions of the pattern into two disjoint subsets. The comparison of characters is carried out using a specific order declared in a matrix. The process requires a pretreatment step of order O(m2) while the search complexity is O(n) in the most complex cases performed in comparison of characters.

11)Apostolico-Giancarlo algorithm [26]: Boyer-Moore algorithm is difficult to analyze because after each search, it does not memorize the characters already found. To remedy this, Apostolico and Giancarlo have designed an algorithm that records the length of the longest suffix of the text that ends at the correct position of the window at the end of each search. The spatial and temporal complexity of this algorithm is similar to that of Boyer-Moore O (m+n). During the search phase, only the last information m of the table break is needed for each attempt so that the size of the table break can be reduced to O (n). The disadvantage of the Apostolico-Giancarlo algorithm is that it happens, in some cases, to perform up to (32n) comparisons of text characters.

12)Raita algorithm [27]: Raita's algorithm was produced by Tim Raita in 1992. The pretreatment phase of the Raita algorithm consists of calculating the bad character shift function (Boyer-Moore). It first compares the last character of the pattern with the rightmost text character of the window, in the case of matching, it continues to compare the first character of the pattern with the leftmost text character of the window. If again the match is found, it makes a comparison between the character of the middle of the pattern with the text character of the middle of the window. Finally, if the match is found, it continues to compare the other characters from the penultimate to the last, eventually by comparing again the character of the medium. During the preprocessing phase, the Raita algorithm requires temporal complexity ($m + n$) and spatial complexity O(n). While in the search phase this algorithm has an extreme quadratic time complexity.

13)Reverse Factor algorithm [28]: The Reverse Factor algorithm results from the use of the smallest inverse pattern suffix automaton, to match some prefixes of the pattern by scanning the character of the window from right-to-left and improving the shift length. The pretreatment phase is linear in time and space. During this phase, the algorithm tries to calculate the smallest automaton suffix for the inverse pattern. During the search phase, the Reverse Factor algorithm analyzes the characters of the window from right to left until any the completion of any transition defined for the current character of the window from the current state of the automaton. At this point, it is easy to know which is the longest prefix length of the matched pattern. The Reverse Factor algorithm requires quadratic time complexity in the worst case, but is optimal on average. It performs O (n.log (m) / m) inspections of text character on average.

14)Berry-RavinrASYdran algorithm [29]: This algorithm consists in ensuring shifts by taking into account the bad character shift (Boyer-Moore algorithm) for the two consecutive text characters immediately to the right of the window. In the preprocessing phase, which requires spatial and temporal complexity of order (m+n2), the algorithm attempts to compute for each pair of characters (a, b) with a, b in Σ the occurrence the most to the right of ab. The search step of the Berry-Ravindran algorithm has a time complexity O(m+n).

15)Aho–Corasick algorithm [30]: Aho-Corasick algorithm falls into the category of dictionary matching algorithms because it performs the localization of the elements of a finite set of strings (the "dictionary") in an entered text. This is achieved by ensuring a correspondence to all chains simultaneously. Both the preprocessing phase and the search phase require a complexity of order O (m + n).

16)Alpha Skip Search algorithm [31]: This algorithm uses buckets of positions for each factor of length log$^{\sigma}$(m). The preprocessing phase requires temporal and spatial complexity of order O (m). The worst case of this pretreatment phase is linear if the size of the alphabet is considered a constant. The temporal complexity of the search phase in the worst case is quadratic, but the expected number of text character comparisons is O(log$^{\sigma}$(m).(n / (m-log$^{\sigma}$(m)))).

B. Comparative Study of the String Matching Algorithms

Iji and Mahalakshmi [6] performed a comparative study of the aforementioned algorithms (Table I) using the sequence JN222368 (Genbank) with a size of 3481 characters. In case of a larger or smaller sequence size the process and the results in terms of accuracy do not change in contrast to the execution time which proportionally depends on the size of the sequences. The tests were conducted using the online tools EMBOSS and GENE Wise.

The results of this study revealed that the Boyer-Moore (BM) chain matching algorithm provides the highest accuracy, 83%, with an execution rate of about 84 ms. The Reverse Colussi (RC) chain matching algorithm provides the shortest execution time (≈57 ms) with an accuracy of 79%. To prove the performance of the proposed approach, DTC was tested with the two best algorithms BM and RC. The experimental results of this test are presented in Section IV.

TABLE I. COMPARATIVE STUDY OF STRING MATCHING ALGORITHMS

Algorithm	**complexity**	**Accuracy**	**Execution Time**
Brute Force	O(mn)	66.7%	≈ 85ms
Deterministic Finite Automaton	O(n)	72%	≈ 65ms
Rabin-Karp	O(mn)	70%	≈ 72ms
Morris-Pratt	O($n + m$)	65%	≈ 68ms
Colussi	O(n)	74%	≈ 58ms
Boyer-Moore	O(mn)	83%	≈ 84ms
Turbo-BM	O(mn)	82.52%	≈ 86ms
Tuned Boyer-Moore	O(mn)	82.1%	≈ 88ms
Reverse Colussi	O(n)	79%	≈ 57ms
Apostolico-Giancarlo	O(n)	74%	≈ 61ms
Smith-Waterman	O(mn)	71.4%	≈ 81ms
Needleman–Wunsch	O(mn)	60%	≈ 85ms
Raita	O(mn)	76%	≈ 82ms
Reverse Factor	O(mn)	75.4%	≈ 82ms
Berry-Ravindran	O($m + n$)	77%	≈ 74ms
Aho–Corasick	O($m + n$)	79.7%	≈ 70ms
Alpha Skip Search	O(mn)	78.5%	≈ 83ms

III. DTC ALGORITHM

Unlike the aforementioned algorithms, which attempt to find a point-by-point correspondence of strings, the DTC approach addresses this problem in its entirety by performing a superposition of the discrete representation of the test points on the continuous representation of the reference points. In this section, the principle of operation of the algorithm will be presented in detail:

Given two sets of points, F= $\{f_i \in R^d\}_{i=1}^{n}$ (the model set) and $SF = \{Sf_j \in R^d\}_{j=1}^{m\leq n}$ (the data set) in the multidimensional

space (ND)

For this application, SF and F respectively represent the DNA sequence in question and the sequence of references. The said sequences are composed of nucleotides in the form of the alphabets (nucleotide) T, C, A and G.

The alignment attempts to find the correspondences between these two points clouds to compare. In this work, we propose an implementation of DTC, as a method of alignment of DNA sequences according to mathematical metrics. Each nucleotide of F and SF is represented by an abscissa (position in the sequence) and an ordinate (a code corresponding to the type of the nucleotide). For our application case, the nucleotides were assigned the following codes: (A = 200, C = - 200, T = 400 and G = -400).

Generally, to decide if the form SF is included in the form F (Fig. 1), one starts by finding a point-by-point correspondence between SF and F and possibly looking for a transformation which would superimpose SF on F.

In the case where one opts for the search for the transformation T (which checks the superposition of SF in F), a direct search of the latter, without any prior knowledge of the correspondence of the SF points with respect to those of F, may be very consuming in terms of execution time caused by the large number of possibilities to test (combinatorial explosion).

It should be noted that the existence of such a transformation T would obviously confirm the inclusion of SF in F. In this respect, the DTC algorithm has the ultimate goal of finding the transformation T in order to confirm the inclusion of SF in F.

Recalling that the origin of the difficulty (combinatorial explosion) comes from the discrete nature of the clouds of points to be treated. The solution proposed by DTC to avoid this problem is to make a transition from the discrete representation to continuous representation of one of the entities (F).

In this case, with a continuous representation of F by a polynomial interpolation (SF would be retained in its discrete representation), the problem of deciding if SF is included in F thus becomes the research, not of T, but at first of a transformation T’ which would bring back SF, on the continuous representation of F.

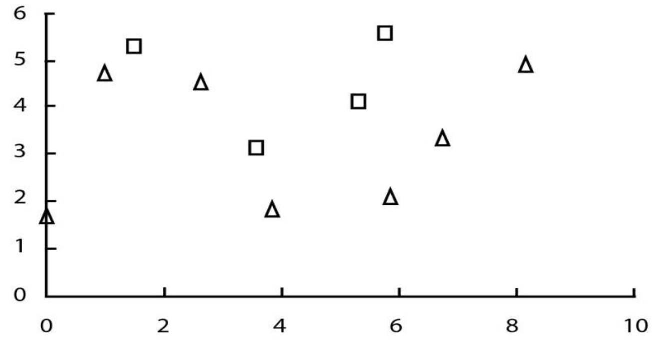

Fig. 1. Alignment of SF on F (Δ:F and □:SF) Search for Transformation T.

Thus the existence of T' could induce the probable existence of T, and therefore, will confirm the inclusion of SF in F.

In fact, the algorithm consists in avoiding a direct search for T but rather in carrying out a search for a transformation T', which would ensure the superposition of SF on the continuous representation of F.

It should be noted that the existence of such a transformation T' is necessary but not sufficient to confirm the inclusion (total or partial) of SF in F. Indeed, the transformation T' (if it exists) must ensure that SF is returned to the continuous representation of F. And if for points$P_{sf_i} \in SF$there exists a point $P_{f_j} \in F$ such that$T'(P_{sf_i}) = P_{f_j}$then T' = T then SF is totally or partially included in F.

The DTC algorithm is developed to deal with arbitrary models defined by cloud of points models in a N-dimensional (ND) space. In the case of our application, the models of the clouds will be considered in a space of 2 dimensions (2D).

The points of F are given in the O_{xy}plane.

Let P_{xy} be the interpolation polynomial in the O_{xy} plane. Because of this, for each point $f_i = \{x_i, y_i\}$belongs to F, we have:

$$P_{xy}(x_i) = y_i \quad (1)$$

This representation will be called (R).

There are different interpolation methods to represent R. In order for the degree of the polynomial to not depend on the size of the point cloud F, the DTC algorithm uses the "cubic spline" interpolation which is a third degree polynomial succession (piecewise interpolation), which also ensures the continuity and the differentiability over the entire interpolation interval (Fig. 2).

Search for transformation T':

The purpose of the transformation T' sought is to bring back the cloud SF on the cubic interpolation of the form F along the plane O_{xy}.

The desired transformation T' is expressed in homogeneous coordinates.

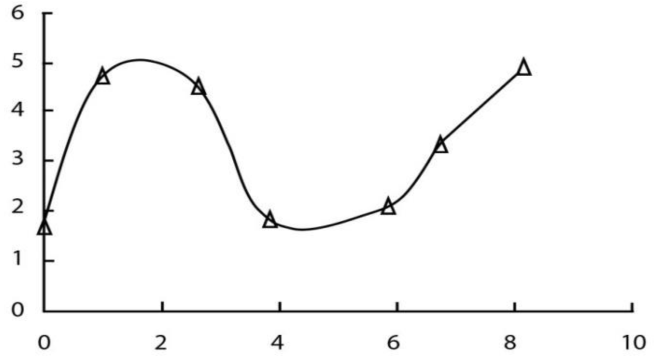

Fig. 2. Cubic Spline Interpolation of F.

If we consider $\left(x_{SF_j}, y_{SF_j}\right)$the coordinates of a point $j \in SF$ and $\left(x'_{SF_j}, y'_{SF_j}\right)$its transformation by T', the transformed points of SF must verify R namely:

$$P_{xy}\left(x'_{SF_j}\right) = y'_{SF_j} \tag{2}$$

The parameters of the transformation T' described in DTC relates to a three-dimensional space are:

Three translations: t_x, t_y, t_z

t_x : Translation along the axisO_x.

t_y : Translation along the axisO_y.

t_z : Translation along the axis O_z.

Three rotations: $\theta_x, \theta_y, \theta_z$

θ_x : Rotation along the axis O_x.

θ_y : Rotation along the axis O_y.

θ_z : Rotation along the axis O_z.

Three scale factors: $\lambda_x, \lambda_y, \lambda_z$

λ_x : Scale factor along the axis O_x.

λ_y : Scale factor along the axis O_y.

λ_z : Scale factor along the axis O_z.

In this work we deal with the problem of alignment of the biological sequences, the transformation T' becomes a function with a single parameter which is the translation t_x according to axis O_x (Fig. 3).

$$x'_{SF_j} = x_{SF_{j+t_x}} \tag{3}$$

$$y'_{SF_j} = y_{SF_j} \tag{4}$$

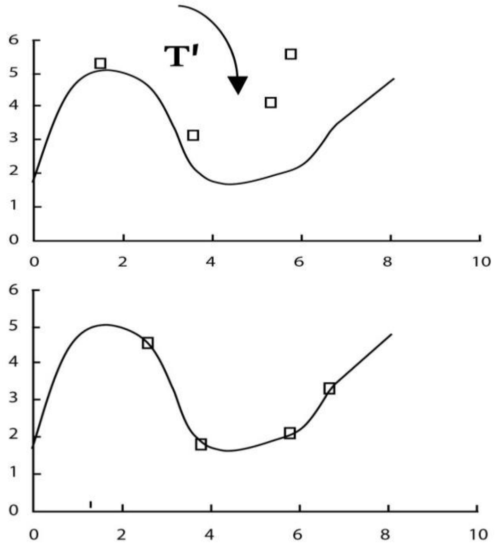

Fig. 3. Illustration for T' Research.

The notation of R can thus be written as:

$$P_{xy}\left(x'_{SF_j}\right) - y'_{SF_j} = 0 \tag{5}$$

Or:

$$\left(P_{xy}\left(x'_{SF_j}\right) - y'_{SF_j}\right)^2 = 0 \tag{6}$$

Applied to all SF points:

$$\sum_{j=1}^{m} \sqrt{\left(\left(P_{xy}\left(x'_{SF_j}\right) - y'_{SF_j}\right)^2\right)} = 0 \tag{7}$$

Consider QT as the following expression:

$$QT_{(t_y)} = \sum_{j=1}^{m} \sqrt{\left(\left(P_{xy}\left(x'_{SF_j}\right) - y'_{SF_j}\right)^2\right)} \tag{8}$$

Based on this definition, we now look for the parameters of the transformation T' which minimizes the QT function.

The obtained function QT is a non-linear equation which is continuous and differentiable.

After the step of adjusting the points of SF on the continuous representation of F (defined by T'), we associate each point of SF with its isomorph, which is its nearest neighbor in F according to a type of distance and a predetermined threshold (ε) (Fig. 4(a)).

$P_j \in SF$: $T\left(x_{SF_j}, y_{SF_j}\right) = \left(x''_{SF_j}, y''_{SF_j}\right)$and if $P_i\left(x_{F_i}, y_{F_i}\right)$ is the isomorph of the point P_j of SF defined by T (Fig. 4(b)).The Root Mean Score (RMS) which is used to measure the global precision of the superposition of SF in F is:

$$RMS = \frac{1}{m} \sqrt{\sum_{j=1}^{m} \left(x''_{SF_j} - x_{F_i}\right)^2 + \left(y''_{SF_j} - y_{F_i}\right)^2} \tag{9}$$

At this stage, since the isomorphs of the points of SF in F are known, it would be possible, if necessary, to refine the superposition.

RMS is also formulated as a non-linear function. Generally, the transformation T thus found, provides directly $T'$and therefore the solution of the RMS is already found (Fig. 4).

Nevertheless, some refinements can be operated according to a predefined RMS threshold. After determining the parameter of the RMS(t'_y), only the points of SF whose distance to their isomorphs in F, are less than a threshold ε fixed in advance. If all the distances between the points of SF to their isomorphs in F, do not exceed ε, then SF is declared included in F. If SF is declared not included in F, the DTC algorithm can process the Largest Common Point Set (LCP) between the two sequences. Indeed, the isomorphic pairs which do not respect the predefined threshold ε would be eliminated, and a revival of a refinement of RMS will take place for a better superposition of the remaining points of SF on the points of F.

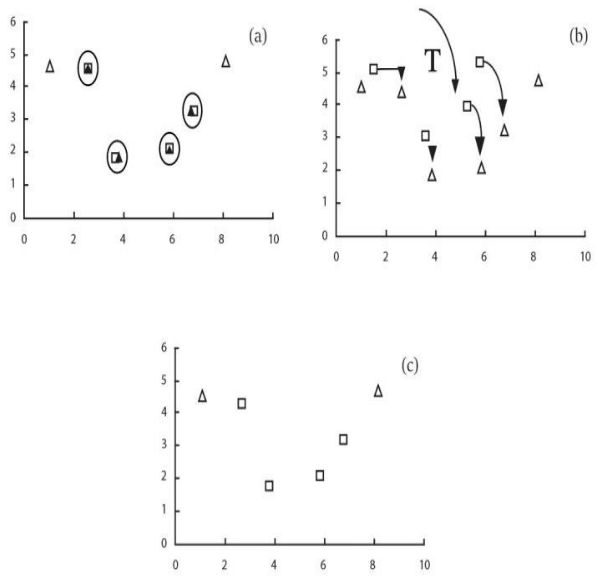

Fig. 4. a: Illustration for Isomorphs Research/b: Illustration for T Research /c: Illustration for Calculation of the Corresponding RMS.

IV. Experimental Results

As mentioned in Section III, a survey of exact string matching algorithms for motif detection in the protein sequence was performed. The results of this comparative study are shown in Table I. Following this analytical study, the aforementioned algorithms were analyzed in terms of time complexity, response time and accuracy using online tools such as EMBOSS and GENE Wise. The sequence studied in this work is JN222368 for Genbank belonging to the marine sponge. Experimental results revealed that the Boyer-Moore (BM) chain matching algorithm provided the highest accuracy 83% with a run rate of about 84 ms. The Reverse Colussi (RC) chain matching algorithm provides the shortest execution time (≈57 ms) with an accuracy of 79%. These results have sparked the interest in implementing the DTC algorithm to test it in terms of execution time and accuracy. To do this, the test environment and conditions were unified by downloading a partial Genbank database containing 7682 sequences of different sizes including our sequence of interest JN222368. Subsequently, the DTC algorithm has been implemented as well as the other two reference algorithms Boyer-Moore and Reverse Colussi in the Java programming language. The machine used was a 2.40 GHz Intel Core i7 processor with 8 GB of RAM. Table II presents the results of comparison of DTC approach with the aforementioned algorithms.

Where m represents the size of F and n represents the size of SF.

TABLE II. Comparison of DTC with Boyer Moore and Reverse Colussi

Algorithms	Execution Time	Time Complexity
(BM)	74 ms	O(mn)
(RC)	51ms	O(n)
(DTC)	42ms	O (m log (n))

For this type of test, the three algorithms have an accuracy of 100%. This is an excellent result for ensuring alignment in restricted databases as an example for the deployment in mutation prediction software, stored on a local server, which will make it possible to compare them with the new sequenced genomes. However, to increase the challenge in terms of accuracy, it would be wise to perform tests (Big Data) by accessing online databases.

As far as temporal complexity is concerned, the proposed algorithm has a log complexity O (m log (n)), unlike BM O (m + n) and RC O (m2). This explains the reduced response time of DTC approach (42 ms), compared to other BM (74 ms) and RC (51ms) algorithms. This means that our algorithm is faster than all 18 algorithms that were the subject of that aforementioned study. Another advantage that has led to this performance in terms of processing time consists in the possibility of storing the interpolations of the reference sequences. This practice has allowed us to save preprocessing time. To demonstrate the temporal complexity of DTC, a test was also performed. It consists in finding an alignment of the form SF on the reference form F (JN222368) with different sizes of SF. The response time and the success rate of this test are shown in Table III.

In this case, for the different sizes of SF, the response time follows a logarithmic evolution (Fig. (5).

This logarithmic complexity makes DTC more efficient in terms of response time in the processing of long sequences. As the results of this test show, the processing time of 600 characters is the same for 3481 characters (15 ms).

In the contrary of our algorithm, BM and RC fail to detect mutations and Gaps. To determine the performance of DTC in detecting mutations and Gaps, tests were performed on the same sequence (with a size of 3481) by simulating mutations (Table IV) and gaps (Table V). The search was carried out as indicated above in a database containing 7682 other sequences of different sizes.

Mutation test: To simulate mutations, nucleotide modifications were made (5 to 60% of the modifications) on our sequence of interest. The results of this test are shown in Table IV.

The results of this test revealed that up to a mutation rate of 60% the algorithm remains insensitive to mutations and the variation of the response time remains marginal despite the considerable change in the rate of mutations. The change choice of 60% is beyond this rate of change, it would no longer be a mutation, but another problem for which the algorithm provides another solution.

Gap test: The representation of the DNA sequences is a succession of the alphabets A, C, G and T. For the simulation of the gaps some SF nucleotides will be replaced by a letter x (unknown) which, in F, would be isomorphic to A, C, G or T. The gap phenomenon is often encountered in sequence alignment and some algorithms find it difficult to treat it (such as Boyer Moore and Reverse Collusi).

DTC approach is still very efficient in terms of gap treatment because any gap corresponds to the reduction of the size of the SF sub-sequence, which generates a reduction in processing time when the gap rate increases.

As shown in the table, the processing of the sequence with 5% gaps lasted longer (25 ms) than with 60% gaps (18 ms).

This explains the performance of DTC in solving this phenomenon often encountered during DNA sequencing.

TABLE III. RESPONSE TIME AND DTC SUCCESS RATE FOR ALIGNMENT OF DIFFERENT SF SIZES ON F

Size of SF	Success rate %	Response time (ms)
60	100 %	10
120		12
600		15
1000		17
2000		24
3000		16
3481		15

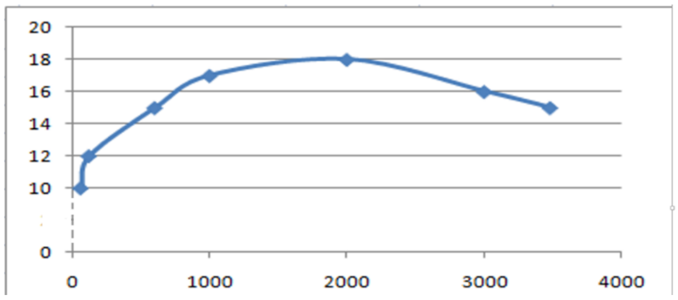

Fig. 5. Average Run Time for each Size of SF.

TABLE IV. SUCCESS RATE OF ALIGNMENT AND RESPONSE TIME FOR THE CASE OF MUTATIONS

Mutation rate %	Response time (ms)
5	24
15	27
25	25
35	27
45	29
50	30
60	31

TABLE V. SUCCESS RATE OF ALIGNMENT AND RESPONSE TIME FOR THE CASE OF GAPS

Gap rate %	Response time (ms)
5	25
15	23
25	23
35	23
45	22
50	21
60	18

V. CONCLUSION

The field of analysis and interpretation of DNA sequences is essential for determining the functional and structural relationships of said sequences. To do this, software based on intelligent algorithms has been made available to the scientific community. In this work, we have presented and compared the DTC algorithm based on polynomial interpolation with algorithms commonly used in this field of application namely: Boyer Moore and Reverse Collussi. The peculiarity of DTC algorithm is that it ensures the exact string matching and the approximate matching with a very short response time, avoiding the preprocessing time by storing the interpolations of the reference sequences. The satisfactory results of the proposed approach, encourage to realize our own software for the detection of gene mutations predisposing to various genetic diseases.

On the other hand, it is possible to apply the DTC algorithm to facilitate and accelerate the implementation of metagenomic analysis as a tool for rapid and precise diagnosis or prognosis of cancers. The metagenomic approach allows us to gain a fairly precise understanding of the molecular mechanisms at work in the emergence and progression of cancer. The application of DTC will make it possible to identify relevant bioindicators / biomarkers (bacterial taxa / genes or metabolic profiles) in order to be able to propose diagnostic and therapeutic approaches for the population-specific. The expected potency of DTC plus the "exhaustive" aspect of metagenomics would also allow the discovery of new genes or biotechnological functions.

ACKNOWLEDGMENTS

We would like to thank the Technical Support Units for Scientific Research (MARWAN) of the National Center for Scientific and Technical Research (CNRST) in Rabat, Morocco for using the HPC platform.

AUTHORS' CONTRIBUTIONS

All authors are equally contributed in this work and this paper.

Wajih Rhalem: Participated in all experiments, coordinated the data-analysis and contributed to the writing of the manuscript.

Jamal El Mhamdi: Coordinated the mouse work, designed the research plan and organized the study.

Mourad Raji: Participated in all experiments, coordinated the data-analysis and contributed to the writing of the manuscript

Ahmed Hammouch: Coordinated the mouse work, designed the research plan and organized the study.

Aqili Nabil: Participated in all experiments, coordinated the data-analysis and contributed to the writing of the manuscript.

Nassim Kharmoum: Participated in all experiments, coordinated the data-analysis and contributed to the writing of the manuscript.

Hassan Ghazal: Participated in all experiments, designed the research plan, coordinated the data-analysis and contributed to the writing of the manuscript

ETHICS

This article is original and contains unpublished material. The corresponding authors have read and approved the manuscript and no ethical issues involved.

REFERENCES

[1] B. Needleman, Saul, Wunsch and D. Christian, "A general method applicable to the search for similarities" in the amino acid sequence of two proteins. Journal of Molecular Biology, vol. 48, pp. 443–453, 1970.

[2] T. F. Smith and M. S. Waterman, "Identification of Common Molecular Subsequences", Journal of Molecular Biology, Vol. 147, pp:195–197. 1981.

[3] M. Raji and A. Cossé-Barbi, "Shape recognition and chirality measure: reestablishing the link between similarity and dissimilarity in discrete space. Chemometrics and intelligent laboratory systems, vol. 47, pp. 219_225, 1999.

[4] W. Rhalem, M. Raji, A. Hammouch and J. El Mhamdi, "An automated time-shift alignment algorithm based on Discret to continuous approach". Journal of Computer science, vol. 15, pp. 463-474, 2019.

[5] W. Rhalem, J. El Mhamdi, M. Raji, A. Hammouch, Abd-ErrahimMaazouzi , S. Raoui , S. Amzazi , S. Hamdi and H. Ghazal Application of a discrete to continuous approach based -alignment algorithm for Capillary Electrophoresis DNA sequencing correction. Advances in Intelligent Systems and Computing-Springer Book. vol. 4 pp.141-148, 2019.

[6] N. Aqili, A. Maazouzi, M. Raji, A. Jilbab and S. Chaoukiet al., "On-line signature verification using point pattern matching algorithm" Proceedings of the International Conference on Electrical and Information Technologies, May 4-7, IEEE Xplore Press, Tangiers, Morocco, pp: 410-413, 2016.

[7] A. Maazouzi, N. Aqili, M. Raji and A. Hammouch, "A speaker recognition system using power spectrum density and similarity measurements", Proceedings of the 3rd World Conference Complex Systems, Nov. 23-25, IEEE Xplore Press, Marrakech, Morocco, pp: 1-5, 2015.

[8] N. Aqili, A. Hammouch and M. Raji, "PPM translation, rotation and scale in d-dimensional space by the discrete to continuous approach". Int. Rev. Comput. Softw., vol.11, pp. 270-276, 2016.

[9] N. Aqili, A. Maazouzi, M. Raji, A. Jilbab and A. Hammouch, "Fingerprint matching algorithm based on discrete to continuous approach. Proceedings of the International Conference on Electrical and Information Technologies", IEEE Xplore Press, Tangiers, Morocco, pp: 414-417, 2016.

[10] N. Nadia Ben, T. Lecroq and M. Elloumi, "A fast Boyer-Moore type pattern matching algorithm for highly similar sequences". Int. J. Data Mining and Bioinformatics, vol. 13, pp. 266-288. 2015.

[11] R. Beal and D. Adjerh "Efficient pattern matching for RNA secondary structures". Theoretical Computer Science. Vol. 592, pp 59-71, 2015.

[12] N. Iji and T. Mahalakshmi, "Survey of Exact String Matching Algorithm for Detecting Patterns in Protein Sequence". Advances in Computational Sciences and Technology. vol. 10, pp: 2707-2720, 2017.

[13] E. Rucci, C. Garcia Sanchez, G. Botella Juan, A.D. Giusti, M. Naiouf and M. Prieto-Matias, "SWIMM 2.0: Enhanced Smith–Waterman on Intel's Multicore and Manycore Architectures Based on AVX-512 Vector Extensions". International Journal of Parallel Programming, vol 47, pp. 296-316, 2019.

[14] S. Kouchaki, A.Tapinos and D. L. Robertson, "A signal processing method for alignment-free metagenomic binning: multi-resolution genomic binary patterns". Scientific Reports, vol. 9, 2019.

[15] C. Ryu, T. Lecroq and K. Park, "Fast string matching for DNA sequences". Theoretical Computer Science, vol. 812, pp. 137-148, 2020.

[16] M. Karp, Richard, Rabin and O. Michael, "Efficient randomized pattern-matching algorithms". IBM Journal of Research and Development, vol. 31, pp. 249–260. 1987.

[17] E. Rasywir, Y. Pratama, Hendrawan and M. Istoningtyas, " Removal of modulo as hashing modification process in essay scoring system using rabin-karp", International Conference on Electrical Engineering and Computer Science, ICECOS pp. 159-164, 2018.

[18] L. Yehia and C. Eng, "Largescale population genomics versus deep phenotyping: Brute force or elegant pragmatism towards precision medicine" npj Genomic Medicine, vol. 4., 2019.

[19] A. Kumar, M. Singh and A.R. Pais, "Fuzzy string matching algorithm for spam detection in twitter Communications" in Computer and Information Science, vol. 939, pp. 289-301, 2019.

[20] R. S. Boyer and J.S, Moore, "A Fast String Searching Algorithm". Comm. ACM. New York, NY, USA, Association for Computing Machinery, vol. 2, pp. 762–772, 1977.

[21] M. Crochemore, A. Czumaj, L. Gasieniec, S. Jarominek, T. Lecroq, W. Plandowski and W. Rytter, Deux méthodes pour accélérer l'algorithme de Boyer-Moore. Théorie des Automates et Applications, Actes des 2e Journées Franco-Belges, D. Krobed., Rouen, France, PUR, vol. 176, pp. 45-63, 1992.

[22] A. Hume and D.M. Sunday, "Fast string searching" Software - Practice & Experience, vol. 21, pp. 1221-1248, 1991.

[23] M. Crochemore and C. Hancart, Automata for Matching Patterns. Handbook of Formal Languages, Linear Modeling: Background and Application, G. Rozenberg and A. Salomaa ed., Springer-Verlag, Berlin, vol. 2, pp. 399-462, 1997.

[24] J.H. Morris, and V.R. Pratt, "A linear pattern-matching algorithm", Technical Report 40, University of California, Berkeley. 1970.

[25] L. Colussi, "Fastest pattern matching in strings", Journal of Algorithms, vol. 16, pp. 163-189, 1994.

[26] A. Apostolico, R.Giancarlo, "The Boyer-Moore-Galil string searching strategies revisited". SIAM Journal on Computing, vol. 15, pp. 98-105, 1986.

[27] T. Raita, "Tuning the Boyer-Moore-Horspool string searching algorithm", Software - Practice & Experience, vol. 22, pp. 879-884, 1992.

[28] T. Lecroq, "A variation on the Boyer-Moore algorithm". Theoretical Computer Science. Elsevier, vol. 92, pp. 119-144,1992.

[29] T. Berry, and S. Ravindran. "A fast string matching algorithm and experimental results". Proceedings of the Prague Stringology Club Workshop`99, J. Holub and M. Simánek ed., Collaborative Report DC-99-05, Czech Technical University, Prague, Czech Republic, pp 16-26, 1999.

[30] V. Aho, Alfred, Corasick and J. Margaret, "Efficient string matching: An aid to bibliographic search", Communications of the ACM, vol. 18, pp.333–340, 1975.

[31] C. Charras, T. Lecroq and J.D. Pehoushek, "A very fast string matching algorithm for small alphabets and long patterns", Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching, M. Farach-Colton ed., Piscataway, New Jersey, Lecture Notes in Computer Science. Springer-Verlag, Berlin, vol. 1448, pp. 55-64, 1998.