Research Article / Review Article FUPRE JOURNAL 9(1):01-13(2025)

FUPRE Journal

of

Scientific and Industrial Research

ISSN: 2579-1184(Print) ISSN: 2578-1129 (Online)

http://fupre.edu.ng/journal

Modeling Meiotic Rearrangements: Using Dynamic Programming to Elucidate the Role of DNA Alignments in Cilliate Reproduction

OSANAKPA, R. O. [1], ***UGBENE, I. J.*** [2,*]

[1,2]*Department of Mathematics, Federal University of Petroleum Resources, Effurun, Delta State*

ARTICLE INFO

Received: 20/11/2024
Accepted: 20/03/2025

Keywords

Algorithm, Cilliate reproduction, Deoxyribonucleic acid (DNA), Genetic factors, Meiotic rearrangement

ABSTRACT

Dynamic programming algorithms are powerful tools for analyzing complex biological data, including DNA sequences. In this study, we employed a combination of global and local alignment algorithms, affine gap and star alignment algorithms, and multiple alignment algorithms to analyze DNA sequences obtained from different ciliates during meiotic reproduction. Our analysis revealed that these algorithms were effective in identifying conserved regions and patterns in the DNA sequences, and in constructing phylogenetic trees that reflected the evolutionary relationships among the sequences. Specifically, we found that the global alignment algorithm was useful for identifying long stretches of identical nucleotides, while the local alignment algorithm was effective in detecting shorter, conserved regions. The affine gap model allowed us to account for the presence of gaps in the sequences, while the star alignment algorithm enabled us to identify conserved regions that were shared among multiple, closely related sequences. Finally, the multiple alignment algorithm allowed us to compare the DNA sequences of multiple ciliates simultaneously, and to identify conserved regions that were shared among all of the species studied. Our findings have important implications for our understanding of the evolution and diversity of ciliates and other organisms, and highlight the utility of dynamic programming algorithms in analyzing complex biological data. Overall, our study provides a framework for using dynamic programming algorithms to analyze DNA sequences, and demonstrates the potential of these algorithms to provide insights into the genetic factors that underlie evolution and diversity in a wide range of organisms.

1. INTRODUCTION

The ciliates are a group of alveolates characterised by the presence of hair-like organelles called cilia, which are identical in structure to eukaryotic flagella, but are in general shorter and present in much larger numbers, with a different undulating pattern than flagella. Cilia occur in all members of the group (although the peculiar Suctoria only have them for part of their life cycle) and are variously used in swimming, crawling, attachment, feeding, and sensation. Ciliates are an important group of protists, common almost anywhere there is water—in lakes, ponds, oceans, rivers, and soils, including anoxic and oxygen depleted habitats(Rotterov´a et al. (2022)). About 4,500 unique free-living species have been described, and the potential number of extant species is estimated at 27,000–40,000, Foissner and Hawksworth (2009). Included in this number are many ectosymbiotic and

*Corresponding author, e-mail: ugbene.ifeanyi@fupre.edu.ng
DIO

endosymbiotic species, as well as some obligate and opportunistic parasites. Ciliate species range in size from as little as 10 µm in some colpodeans to as much as 4 mm in length in some geleide, and include some of the most morphologically complex protozoans, Lynn (2008); Nielsen and Kiørboe (1994). Unlike most other eukaryotes, ciliates have two different sorts of nuclei: a tiny, diploid micronucleus (the" generative nucleus", which carries the germline of the cell), and a large, ampliploid macronucleus (the" vegetative nucleus", which takes care of general cell regulation, expressing the phenotype of the organism), (Archibald et al. (2017)). The latter is generated from the micronucleus by amplification of the genome and heavy editing. The micronucleus passes its genetic material to offspring, but does not express its genes. The macronucleus provides the small nuclear RNA for vegetative growth, (Archibald et al. (2017); Prescott (1994)). Division of the macronucleus occurs in most ciliate species, apart from those in class Karyorelictea, whose macronuclei are replaced every time the cell divides, (Lynn (2008)). Macronuclear division is accomplished by amitosis, and the segregation of the chromosomes occurs by a process whose mechanism is unknown, (Archibald et al. (2017)). After a certain number of generations (200–350, in Paramecium aurelia, and as many as 1,500 in Tetrahymena (Lynn (2008))) the cell shows signs of aging, and the macronuclei must be regenerated from the micronuclei. Usually, this occurs following conjugation, after which a new macronucleus is generated from the post-conjugal micronucleus, (Archibald et al. (2017)). Pevzner et al. (2001) introduces the concept of using Eulerian paths to address the challenge of DNA fragment assembly. The authors propose an algorithm that reconstructs the original DNA sequence from a collection of short DNA fragments. While not directly related to ciliate reproduction, this work highlights the importance of computational methods in understanding and analyzing DNA rearrangements. Dobzhansky (1933)'s study on the sterility of interracial hybrids in Drosophila pseudoobscura provides insights into the genetic basis of reproductive isolation and speciation. Although not specific to ciliates, this article emphasizes the role of genetic incompatibilities resulting from DNA rearrangements in reproductive barriers and evolutionary processes. Orr (1996) explores the contributions of Dobzhansky and Bateson to our understanding of genetic mechanisms underlying speciation. The author emphasizes the role of genetic changes, including DNA rearrangements, in driving reproductive isolation and the formation of new species. This work provides a broader context for understanding the evolutionary implications of DNA rearrangements in ciliate reproduction. Pevzner and Tesler (2003) investigates genome rearrangements in mammalian evolution, specifically focusing on the human and mouse genomes. By studying large-scale DNA rearrangements, the authors shed light on the evolutionary processes that shape the organization of genomes. This research provides insights into the broader implications of DNA rearrangements in the context of evolutionary biology. Biller et al. (2016) addresses the challenge of estimating rearrangement distances between genomes, considering the fragility of specific genomic regions. By accounting for fragile regions that are prone to DNA rearrangements, the authors propose an improved method for accurately measuring the distance between genomes. This work highlights the importance of considering the structural properties of genomes in the study of DNA rearrangements.

Yancopoulos et al. (2005) presents an efficient algorithm for sorting genomic permutations, which involve rearrangements such as translocations,

inversions, and block interchanges. While not specific to ciliate reproduction, this work contributes to the computational methods used to analyze DNA rearrangements and can inform our research on modeling meiotic rearrangements in ciliates. Beermann (1977)'s study on the diminution of heterochromatic chromosomal segments in Cyclops provides insights into the DNA rearrangements that occur during the development of this crustacean. The research highlights the process of segment elimination, which involves the removal of specific chromosomal segments during the formation of somatic cells. Although not specific to ciliates, this work contributes to our understanding of DNA rearrangements in different organisms. Gerbi (1986) explores the unusual chromosomal movements in the spermatocytes of sciarid flies. These flies exhibit complex and dynamic chromosomal movements during meiosis, including the formation of chromosomal bouquets and the movement of chromosomes to specific regions of the nucleus. They suggest that these movements may play an essential role in ensuring proper chromosomal segregation during meiosis. The paper provides a detailed analysis of the chromosomal movements in sciarid flies and offers valuable insights into the mechanisms that govern chromosomal segregation in these organisms. Prescott (1994) focuses on the DNA of ciliated protozoa. These organisms have highly complex genomes that undergo programmed genome rearrangements during their life cycle. Prescott provides an overview of the unique features of the DNA of ciliated protozoa, including its structure and organization, and the mechanisms that govern genome rearrangement. The paper also highlights the potential of ciliated protozoa as model systems for studying genome dynamics. Understanding the genome rearrangements in ciliated protozoa can provide insights into the evolution of complex genomes and the mechanisms that drive genome evolution. Smith et al. (2012) discusses the genetic consequences of programmed genome rearrangement in various organisms. The authors describe how these rearrangements can result in the creation of new genes, the deletion of existing genes, and the formation of chimeric genes. They also discuss the potential impact of these rearrangements on genome evolution and the development of new species. The paper highlights the importance of understanding the mechanisms that govern genome rearrangements in order to better understand the evolution of complex genomes. Stephens et al. (2011) describes a catastrophic event that led to massive genomic rearrangement in a single cancer cell. The authors used whole-genome sequencing to identify more than 10,000 rearrangements in a single cell, which is unprecedented in cancer genomics. The study provides important insights into the mechanisms of genome instability in cancer. Aguileta et al. (2014) investigated the variability of mitochondrial gene order among fungi. Mitochondrial genomes are known to be highly variable in terms of gene order and content, and this study provides a comprehensive analysis of this variability in fungi. The authors found that the gene order is highly variable even among closely related species, and that this variability is due to frequent rearrangements and gene loss. Lang et al. (2014) discovered massive programmed translational jumping in mitochondria. The authors found that mitochondrial ribosomes can skip large regions of the genome during translation, resulting in the production of truncated proteins. This finding challenges the traditional view of mitochondrial translation and provides new insights into the evolution of mitochondrial genomes. Ehrenfeucht et al. (2004) discussed the computation that occurs in living cells during gene assembly in ciliates. The authors provide a comprehensive overview of the molecular mechanisms involved in

gene assembly, and discuss how these mechanisms can be modeled using formal language theory.

2. MATERIALS AND METHODS

New DNA, RNA, and protein sequences emerge from existing sequences rather than being created from scratch by nature. This fundamental principle forms the basis of sequence analysis. If we can establish a connection between a newly discovered sequence and a sequence for which some information (such as structure or function) is already known, it is likely that this known information also applies, to some degree, to the new sequence. We consider any two related sequences to have originated from a shared ancestral sequence during the process of evolution and refer to them as homologous sequences. Seeking sequence similarity is the first step in deducing homology. Determining whether two sequences are similar or not can be challenging when they are lengthy. One must correctly align them in order to determine whether they are comparable. Sequences can experience substitutions, or the replacement of one residue by another, when they evolve from a common ancestor. In addition to substitutions, sequences can collect a number of events of two other types during evolution: insertions, which occur when new residues enter in a sequence in addition to the ones that already exist, and deletions, which occur when some residues disappear. Residues must therefore be permitted to align not just to other residues but also to gaps in order to achieve the best possible alignment between two sequences. An insertion or deletion event is indicated by the existence of a gap in an alignment. Take into consideration, for instance, the next two incredibly short nucleotide sequences, each with just seven residues:

$$x: TACCAGT$$
$$y: CCCGTAA \qquad (1)$$

If gaps in alignments are avoided, there is only one method to align the sequences because they are of the same length:

$$x: T\ A\ C\ C\ A\ G\ T$$
$$y: C\ C\ C\ G\ T\ A\ A$$

(2)

There are numerous alignments that are feasible, though, if gaps are allowed. Specifically, the alignment that follows appears to be far more instructive than the one that comes before it:

$$x: TACCAGT\_\_$$
$$y: C\_CC\_GTAA$$

(3)

The subsequence CCGT may be an evolutionarily conserved region, according to alignment (2.1), which suggests that both x and y may have developed from a shared ancestral sequence that contained the subsequence CCGT in the right places. Here's another alignment that seems plausible:

$$x: T\ A\ C\ C\ A\ G\ T\_\_$$
$$y: \_\ \ \_\ \ C\ C\ C\ G\ T\ A\ A$$

(4)

In what way is alignment (3) superior to alignment (4)? Are certain alignments better than others? In order to respond to these inquiries, we must be able to assess every potential alignment. The best or most ideal alignments are therefore those with the highest score (although there may be multiple of these alignments).

The most common scoring techniques make the assumption that each column in an alignment is independent of the others and assign the alignment's overall score to equal the sum of the scores of its individual columns. With a,b ∈ Q, where Q is either the 4-letter DNA or RNA alphabet or the 20-letter amino acid alphabet, depending on the type of sequences that we are interested in aligning, one only needs to specify the scores $s(a,b) = s(b,a)$ and the gap penalty $s(-,a) = s(a,-)$, for such schemes. Naturally, the scoring system affects which

alignments between two sequences are optimal. The optimal alignments for two distinct scoring systems could be very different from one another. One can set $s(a,a) = 1$(the match score), $s(a,b) = -1$ if a=b (the mismatch score), and s(−,a)=s(a,−)= −2 (the gap penalty) as an example of a scoring scheme. It is crucial to remember, nevertheless, that for a scoring system to result in a logical alignment, it must be biologically relevant. A strategy like this needs to account for the constraints on sequence evolution. For instance, by making the score of a continuous gap region an affine function of its length (notice that in the example above, the score of a gap region is linear in its length), many popular scoring schemes establish some degree of dependence among the columns in an alignment. A substitution or scoring matrix is formed by the numbers $s(a,b)$. For sequence comparison to be effective, substitution matrices must be symmetric and have a few other requirements.

2.1 Dynamic Programming: Global Alignment

The linear gap model ($s(-,a) = s(a,-) = -d$ for $a \in Q$, with $d > 0$, is assumed in this section so that the score of a gap area of length L equals $-dL$ and propose the Needleman and Wunsch (1970) algorithm, which is able to identify all optimal global alignments in an algorithmic fashion (there are often multiple such alignments).The goal is to take subsequences that have optimal alignments and turn them into an ideal alignment. Dynamic programming algorithms are commonly defined as algorithms that accomplish optimization by means of executing optimization for smaller bits of data (in this example, subsequences). Two sequences, $x = x_1x_2\ldots x_i\ldots x_n$ and $y = y_1, y_2\ldots y_j\ldots y_m$, are assumed to be. We build a matrix $F = (n+1)\times(m+1)$. The ideal alignment score between $x1\ldots xi$ and $y1\ldots yj$ is represented by its (i,j)th element $F(i,j)$ for $i = 1,\ldots,n, j = 1,\ldots,m$. The score of aligning $x_1\ldots x_i$ to a gap region of length i is represented by the element $F(i,0)$ for $i = 1,\ldots,n$. Similarly, the alignment score of $y_1\ldots y_j$ to a gap region of length j is represented by the element $F(0,j)$ for $j = 1,\ldots,m$. After recursively initializing $F(0,0) = 0$ and filling the matrix from the top left corner to the bottom right corner, we construct F. Upon knowing $F(i-1,j-1), F(i-1,j)$ and $F(i,j-1)$, it is evident how to compute $F(i,j)$:

$$F(i,j) = max \begin{cases} F(i-1,j-1) + s(xi,yj), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{cases} \quad (5)$$

The best score $F(i,j)$ can be attained in fact in three ways: xi has three possible alignments: to a gap (the second option), to y_j (the third option), or to x_i (refer to the first option in the formula above). After computing $F(i,j)$,we maintain a reference to the option that yielded $F(i,j)$. We trace back the pointers to retrieve optimal alignments when we arrive at $F(n,m)$. Their score is precisely $F(n,m)$. Keep in mind that a given matrix cell may produce many pointers, leading to multiple ideal alignments.

2.2. Dynamic Programming: Local Alignment

Finding all pairings of subsequence of two given sequences with the highest-scoring alignments is an alignment problem with more biological interest. Only subsequences of succeeding parts or segments will be of interest to us. Such a subsequence of a sequence$x_1x_2\ldots x_n$ has for each $1 \le i \le n$ and $k \le n-i$ the form $x_ix_{i+1}\ldots x_{i+k}$. We refer to this alignment issue as the ”local alignment problem.” The solution for a linear gap model is provided here using the Smith and Waterman (1981) method. The method for creating a $(n+1)\times(m+1)$-matrix is the same as in the preceding section, but the formula for its entries is varied slightly:

Selecting the first option in the formula above corresponds to initiating a new alignment: it is preferable to initiate a new alignment rather than prolong the existing one if an optimal alignment up to a certain point has a negative score.

$$F(i,j) = max \begin{cases} 0, \\ F(i-1,j-1) + s(xi,yj), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{cases} \quad (6)$$

Another distinction is that an alignment might now terminate anywhere in the matrix. Consequently, we search for the maximum elements in the matrix F and begin traceback from there, rather than calculating the value $F(n,m)$ in the bottom right corner of the matrix for the best score. When we reach a cell with value 0, which is the alignment's beginning, the traceback comes to an end.

2.3 Alignment with Affine Gap Model

Assumed in this section is an affine gap model, where for each $d > 0$ and $e > 0$, the score of any gap area of length L equals $-d - e(L-1)$. In this case, the gap extension penalty is denoted by $-e$ and the gap opening penalty by $-d$. To account for the biological reality that initiating a gap region is more difficult than extending it, e is typically designed to be smaller than d. As in the previous section, we will only address a global alignment technique here; however, a local version can be easily produced as well (see also Gotoh (1982)). One $(n+1) \times (m+1)$ matrix and two $n \times m$ matrices are needed for the algorithm. For $i = 1,\ldots,n$ and $j = 1,\ldots,m$, let $M(i,j)$ represent the best alignment score between $x_1 \ldots x_i$ and $y_1 \ldots y_j$, provided that the alignment terminates with x_i aligned to y_j. The alignment score of $x_1 \ldots x_i$ to a gap region of length i is represented by the element $M(i,0)$ for $i = 1,\ldots,n$. Likewise, the alignment score of $y_1 \ldots y_j$to a gap region of length j is represented by the element $M(0,j)$ for $j = 1,\ldots,m$. Furthermore, let $I_x(i,j)$ represent the best alignment score between $x_1 \ldots x_i$ and $y_1 \ldots y_j$ for $i = 1,\ldots,n$ and $j = 1,\ldots,m$ provided that the alignment terminates with x_i aligned to a gap. In conclusion, let $I_y(i,j)$ represent the ideal alignment score between $x_1 \ldots x_i$ and $y_1 \ldots y_j$ for $i = 1,\ldots,n$ and $j = 1,\ldots,m$ provided that the alignment terminates with y_j aligned to a gap.The following recurrence relations result from assuming that an insertion never comes directly after a deletion (unless the deletion occurs at the start of an alignment):

$$M(i,j) = max \begin{cases} M(i-1,j-1) + s(x_i,y_j), \\ I_x(i-1,j-1) + s(x_i y_j), \\ I_y(i-1,j-1) + s(x_i,y_j) \end{cases} \quad (7)$$

$$I_x(i,j) = max \begin{cases} M(i-1,j) - d, \\ Ix(i-1,j) - e \end{cases} \quad (8)$$

$$I_y(i,j) = max \begin{cases} M(i,j-1) - d, \\ I_y(i,j-1) - e \end{cases} \quad (9)$$

After we set $M(0,0) = 0$ to begin the process, we may use these recurrence relations to fill up the matrices $M, I_x,$ and I_y. The option is not considered in computations if, for some i and j, one of the options in the right-hand sides of the recurrence relations is not defined (for example, in the formula for $M(1,2)$, the right-hand side comprises $I_x(0,1)$ and $I_y(0,1)$).

$\max\{M(n,m), I_x(n,m), I_y(n,m)\}$ is the optimal alignment score, and the traceback begins at the element (or elements) that realize this maximum.

2.4 Multiple Alignment

Common properties between a group of sequences are frequently of interest to sequence analysis researchers. Establishing the best multiple alignment for the entire collection is necessary in order to find such

a feature. Similar to when dealing with two sequences, the ability to score any multiple alignment using a scoring method is necessary to create an ideal multiple alignment. Much like with two sequences, the majority of alignment techniques utilize a scoring function of the kind, assuming that each column in an alignment without gaps is independent.

$$S(M) = G(M) + \sum_i \quad s(M_i),$$

(10)

where M indicates a multiple alignment, M_i is the ith column without a gap, $s(M_i)$ is M_i's score, and G is a function for scoring gaps in columns. The typical (but unsatisfactory) techniques for assigning multiple alignments and gaps-free columns are assessed using the "sum of pairs" (SP) function. For a column Mi that does not contain gaps, the SP-score is defined as

$$s(M_i) = \sum_{k<l} \quad s(M_{ki}, M_{li}),$$

(11)

in which all pairs $(M_{ki}, M_{li}), k < l$ elements of M_i are added up, and the scores $s(a, b)$,3 for$a, b \in Q$, originate from a substitution matrix that is utilized to rate pairwise sequence alignments. In order to score gaps, $s(-, a) = s(a, -)$ is frequently defined.for columns with gaps, introducing the relevant SP-score and setting $s(-, -) = 0$. Any method of scoring gap areas in this way is referred to as a linear gap model for multiple alignments. The SP-score has no statistical basis, despite the seeming common sense of adding together all pairwise substitution scores. Pairwise dynamic programming algorithms can be extended

to align $n \geq 3$ sequences once a system for scoring multiple alignments has been established. When faced with several alignment challenges, one is typically interested in subsequently, a generalization of the Needleman-Winsch technique in global alignments. Here, we'll assume a score system that

$$S(M) = \sum_i \quad s(M_I),$$

(12)

adding up all of the alignment's columns, even the ones with gaps in them. We observe that an affine gap model is also available for a multidimensional dynamic programming approach.Assume that we have n sequences: $x_1 = x_1^1 \ldots x_{m1}^1, x_2 = x_1^2 \ldots x_{m2}^2, \ldots, x_n = x_1^n \ldots x_{mn}^n$. For each integer $i_1, \ldots, i_n, j = 1, \ldots, n,$ where at least one number is non-zero, we have $0 \leq i_j \leq m_j$. Give $F(i_1, \ldots, i_n)$ the maximal score of an alignment of the subsequences that terminate in $x_{i1}^1 \ldots x_{in}^n$ (the other subsequences are aligned to a gap region if for any j we have $i_j = 0$. The dynamic programming algorithm's recursion stage can be found via where all combinations of gaps occur except the one where all residues are replaced by gaps. The algorithm is initialized by setting$F(0, \ldots, 0) = 0$. Traceback starts at $F(m_1, \ldots, m_n)$ and is analogous to that for pairwise alignments. The matrix $F(i_1, \ldots, i_n)$ with $0 \leq i_j \leq m_j, j = 1, \ldots, n,$ is an $(m_1 + 1) \times \ldots \times (m_n + 1)$-matrix, and it is convenient to visualize it by considering its two-dimensional sections.

2.5 MSA

Based on the multi-dimensional dynamic programming algorithm, MSA always identifies all optimal alignments. Reducing the amount of components in the dynamic programming matrix that must be inspected in order to identify the best multiple alignment is the goal. Protein sequences up to 300 residues long can be properly aligned using MSA. Since we are assuming an SP-scoring scheme with a linear gap model for both residues and gaps in this instance, the multiple alignment's score is the total of all the pairwise alignments that the multiple alignment causes. Let k^{th} and

l^{th} sequences be denoted by M_{kl}, and let M represent a multiple alignment. Next up, we have

$$S(M) = \sum_{k<l} \quad S(M_{kl}),$$

(14)

where the M_{kl} score is denoted by$S(M_{kl}).S(M_{kl}) \leq s_{kl}$, evidently, if s_{kl} is the score of an ideal global alignment between the kth and lth sequences.Let us assume that we have a lower bound τ for the ideal multiple alignment score. Any heuristic multiple alignment approach, like the Star Alignment algorithm that is covered below, can find such a bound quickly and with some degree of precision. Consequently, we have for an ideal multiple alignment M_0

$$\tau \leq S(M_0) = \sum_{k^1<l^1} \quad S(M_{kl0}) - s_{kl} + \sum_{k^1<l^1} \quad s_{k^1l^1},$$

(15)

for all k and l. Hence $S(M_{k^1l^10}) \geq t_{kl}$, where

$$t_{kl} = \tau + s_{kl} - \sum_{k^1<l^1} \quad s_{k^1l^1}.$$

(16)

By calculating the scores of the pairwise optimal alignments in the right-hand side of the formula above, as was covered in the sections before, it is possible to compute t_{kl}, at least roughly. Therefore, all that is required of us is to search for such multiple alignments that produce pairwise alignments with scores at least equal to t_{kl}. By significantly reducing the number of elements in the multi-dimensional dynamic programming matrix that require examination, this insight leads to a boost in computational speed.

2.6 Star Alignment

A quick and efficient heuristic technique for generating several alignments is the Star Alignment algorithm. Naturally, it cannot ensure that an ideal alignment will be found, just like any other heuristic approach. The fundamental idea is to use the sequence that most closely resembles all the other sequences as the center of a ”star” that aligns all the other sequences with it.

3 RESULT

In this section, we will employed dynamic programming algorithms to analyze DNA samples obtained from different ciliates during meiotic reproduction. Specifically, we utilized five distinct dynamic programming schemes, namely global alignment, local alignment, affine gap model, star alignment, and multiple alignment, to model the DNA sequences and identify patterns and similarities among the samples.

3.1 Global Alignment with the Needleman-Wunsch algorithm

Let x = CTTAGA, y = GTAA, and suppose that we are using the scoring scheme: $s(a,a)=1$, $s(a,b)=-1$, if $a \neq b$, and $s(-,a)=s(a,-)=-2$. The corresponding matrix F with pointers is derived

	_	G	T	A	A
_	0	-2←	-4 ←	-6←	-8←
C	-2 ↑	-1 ↖	-3 ↖ ←	-5 ↖ ←	-7 ↖ ←
T	-4↑	-3 ↖ ↑	0 ↖	-2 ←	-4 ←
T	-6↑	-5 ↑ ↖	-2 ↖ ↑	-1 ↖	-3 ↖ ←
A	-8↑	-7 ↑ ↖	-4 ↑	-1 ↖	0 ↖
G	-10↑	-7↖	-6 ↑	-3↑	-2↑
A	-12↑	-9 ↑	-8 ↑ ↖	-5 ↑ ↖	-2 ↖

Tracing back the pointers gives the following three optimal alignments

x: CTTAGA
y: G_TA_A

x: CTTAGA
y: GT_A_A

x: CTTAGA
y: _GTA_A

With score −2. The corresponding paths through the matrix *F* are shown with arrows.

Again let, x= CCATACGA, y= CAG C T A G C G, and suppose that we are using the scoring scheme:

$s(a,\ a)$=1, $s(a,\ b)$=−1, if $a \neq b$, and $s(-,a)$=$s(a,-)$= -1.

	–	C	C	A	T	A	C	G	A
–	0	-1	-2	-3	-4	-5	-6	-7	-8
C	-1	1	0	-1	-2	-3	-4	-5	-6
A	-2	0	0	1	0	-1	-2	-3	-4
G	-3	-1	-1	0	0	-1	-2	-1	-2
C	-4	-2	0	-1	-1	-1	0	-1	-2
T	-5	-3	-1	-1	0	-1	-1	-1	-2
A	-6	-4	-2	0	-1	1	0	-1	0
G	-7	-5	-3	-1	-1	0	0	1	0
C	-8	-6	-4	-2	-2	-1	1	0	0
G	-9	-7	-5	-3	-3	-2	0	2	1

This gives ;

x	_	C	A	G	C	T	A	G	C	G	_
y	C	C	A	_	_	T	A	_	C	G	A

3.2 Local Alignment with Smith-Waterman algorithm

For the sequences x = CTTAGA, y = GTAA, the local alignment becomes;

	_	G	T	A	A
_	0	0	0	0	0
C	0	0	0	0	0
T	0	0	↖ 1	0	0
T	0	0	↖ 1	↖0	0
A	0	0	0	↖2	↖1
G	0	↖ 1	0	↑ 0	↖ 1
A	0	0	↖ 0	↖1	↖1

The only best local alignment is:

$$x \quad : T\ A$$
$$y \quad : T\ A$$

and its score is equal to 2, where the arrows represent traceback. Note that if an element of F is equal to 0 and no arrows come out of the cell containing this element, then the element is obtained as the first option in formula (6). Considering, x = C C A T A C G A, y = C A G C T A G C G, and suppose that we are using the scoring scheme: $s(a,a) = 1, s(a,b) = -1$, if $a \neq b$, and $s(-,a) = s(a,-) = -1$.

	–	C	C	A	T	A	C	G	A
–	0	0	0	0	0	0	0	0	0
C	0	1	1	0	0	0	1	0	0
A	0	0	0	2	1	1	0	0	1
G	0	0	0	1	1	0	0	1	0
C	0	1	1	0	0	0	1	0	0
T	0	0	0	0	1	0	0	0	0
A	0	0	0	1	0	2	1	0	1
G	0	0	0	0	0	1	1	2	1
C	0	1	1	0	0	0	2	1	1
G	0	0	0	0	0	0	1	3	2

The optimal score corresponds to the 3 in the last row, but second to last column. The optimal path results in an alignment with four matching positions. The traceback matrix can be built while computing the alignment matrix, and all paths are halted when a score of zero is reached. For Smith-

Waterman, we typically report just the sub-alignment corresponding to the positive scores. We can report an alignment consisting of just the two sequences.

x	T	A	G	C	G
y	T	A	_	C	G

3.3 Alignment with Affine Gap Model

Let $x = \text{ACGGTAC}$, $y = \text{GAGGT}$, the score of any match be equal to 1, the score of any mismatch be equal to -1, $d = 3$ and $e = 2$. Then we get the dynamic programming matrices;

M	_	G	A	G	G	T
_	0	← -3	← -5	← -7	← -9	← -11
A	-3 ↑	↖ -1	↖ -2	↖ -6	↖ -8	↖ -10
C	-5 ↑	↖ -4	↖ -2	↖ -3	-6	-8
					$I_y(1,3$	$I_y(1,4$
G	-7↑	↖ -4	↖-5	↖-1	↖-2	↖-7
			$I_x(2,1$			$I_y(2,4$
G	-9↑	↖ -6	↖-5	↖-4	↖0	↖-3
				$I_x(3,2$		
T	-11 ↑	↖ -10	↖-7	↖-6	↖-5	↖1
					$I_x(4,3$	
A	-13 ↑	↖ -12	-8	↖-8	↖-7	-4
			$I_x(5,1$		$I_x(5,3$	$I_x(5,4$
C	-15 ↑	↖ -14	-12	↖-9	↖-9	-6

			$I_x(6,1$		$I_x(6,3$	$I_x(6,4$

I_x	G	A	G	G	T
A	-6	-8	-10	-12	-14
	$M(0,1$	$M(0,2$	$M(0,3$	$M(0,4$	$M(0,5$
C	-4	-5	-9	-11	-13
	$M(1,1$	$M(1,2$	$M(1,3$	$M(1,4$	$M(1,5$
G	↑ -6	-5	-6	-9	-11
		$M(2,2$	$M(2,3$	$M(2,4$	$M(2,5$
G	-7	↑-7	-4	-5	-10
	$M(3,1$		$M(3,3$	$M(3,4$	$M(3,5$
T	↑ -9	-8	↑-6	-3	-6
	$M(4,1$	$M(4,2$		$M(4,4$	$M(4,5$
A	↑ -11	↑ -10	↑ -8	↑ -5	-2
		$M(5,2$			$M(5,5$
C	↑-13	-11	↑-10	↑ -7	↑ -4
		$M(6,2$			

I_y	G	A	G	G	T
A	-6	-4	-5	← -7	← -9
	$M(1,0$	$M(1,1$	$M(1,2$		
C	-8	-7	-5	-6	← -8
	$M(2,0$	$M(2,1$	$M(2,2$	$M(2,3$	
G	-10	-7	-8	-4	-5
	$M(3,0$	$M(3,1$	$M(3,2$	$M(3,3$	$M(3,4$
G	-12	-9	-8	-7	-3
	$M(4,0$	$M(4,1$	$M(4,2$	$M(4,3$	$M(4,4$
T	-14	-13	-10	-9	-8
	$M(5,0$	$M(5,1$	$M(5,2$	$M(5,3$	$M(5,4$
A	-16	-15	-11	-11	-10
	$M(6,0$	$M(6,1$	$M(6,2$	$M(6,3$	$M(6,4$
C	-18	-17	-15	-12	-12
	$M(7,0$	$M(7,1$	$M(7,2$	$M(7,3$	$M(7,4$

The arrows and labels indicate from which elements of the three matrices each number was produced (in addition, we draw the vertical and horizontal arrows in the 0th column and 0th row of the matrix M). The thick arrows show traceback; it starts at

$I_x(7,5) = -4$. The corresponding optimal alignment with score -4 is:

x: A C G G T A C
y: G A G G T _ _

There are many other variants of the basic dynamic programming algorithm: for overlap matches, repeated matches, more complex gap models, etc.

3.4 Multiple Alignment

We will find all optimal alignments of the three sequences $x =$ AATC, $y =$ GTC, $z =$ AAG using the following scoring scheme: the score of an alignment is calculated from the scores of its columns M_i's from formula (12); if M_i contains three identical symbols, set $s(M_i) = 2$; if it contains exactly two identical symbols, but no gaps, set $s(M_i)=1$; if it contains three distinct symbols, but no gaps, set $s(M_i)=-1$, if it contains exactly one gap, set $s(M_i) = -2$; if it contains two gaps, set $s(M_i) = -4$. In this case, the indices i_1, i_2 and i_3 correspond to sequences x, y and z respectively.

$F(*,*,0)$	_	G	T	C
_	0	← -4	← -8	← -12
A	↑ -4	↖ -2	← ↖ -6	← ↖ -10
A	↑ -8	↑ -6 ↖	↖ -4	← ↖ -8
T	↑ -12	↑ -10 ↖	↑ -8 ↖	↖ -6
C	↑ -16	↑ -14 ↖	↑ -12 ↖	↑ -10 ↖

$F(*,*,1)$	_	G	T	C
_	-4	-2	← -6	← -10
	$F(0,0,0$	$F(0,0,0$	$F(0,1,0$	$F(0,2,0$
A	-2	1	← -3	← -7
	$F(0,0,0$	$F(0,0,0$	$F(0,1,0$	$F(0,2,0$
A	-6 ↑	-3 ↑	-1 ↖	← -5 ↖
	$F(1,0,0$	$F(1,0,0$	$F(1,1,0$	$F(1,2,0$
T	-10	↑ -7	↑ -5 ↖	↖ -3
	$F(2,0,0$		$F(2,1,0$	
C	-14 ↑	-11 ↑	↖ -9 ↑	↖ -7 ↑
	$F(3,0,0$			$F(3,2,0$

$F(*,*,2)$	_	G	T	C
_	-8	-6	-4	← -8
	$F(0,0,1$	$F(0,1,1$	$F(0,1,1$	$F(0,2,1$
		$F(0,0,1$		
A	-6	-3	-1	← -5
	$F(1,0,1$	$F(1,1,1$	$F(1,1,0$	$F(1,2,1$
	$F(0,0,1$	$F(0,1,1$	$F(0,1,1$	$F(0,2,1$
A	-4	-1	2	← -2
	$F(1,0,1$	$F(1,1,1$	$F(1,1,1$	$F(1,2,1$
		$F(1,0,1$		
T	-8 ↑	-5 ↑	-2 ↑	↖ 0
	$F(2,0,1$	$F(2,1,1$	$F(2,1,1$	
C	-12 ↑	-9 ↑	-6 ↑	↖ -4 ↑
	$F(3,0,1$	$F(3,1,1$		$F(3,2,1$

$F(*,*,3)$	_	G	T	C
_	-12	-10	-8	-6
	$F(0,0,2$	$F(0,1,2$	$F(0,2,2$	$F(0,2,2$
		$F(0,0,2$	$F(0,1,2$	
A	-10	-7	-5	-3
	$F(1,0,2$	$F(1,1,2$	$F(1,2,2$	$F(1,2,2$
	$F(0,0,2$	$F(0,0,2$	$F(1,1,2$	
A	-8	-5	-2	0
	$F(1,0,2$	$F(2,1,2$	$F(2,2,2$	$F(2,2,2$
	$F(2,0,2$	$F(1,1,2$		
		$F(1,0,2$		
T	-6	-3	0	1
	$F(2,0,2$	$F(2,1,2$	$F(2,2,2$	$F(2,2,2$
		$F(2,0,2$	$F(2,1,2$	

C	↑ -10	↑ -7	↑ -4	-1
	$F(3,0,2$	$F(3,1,2$	$F(3,2,2$	$F(3,2,2$
		$F(3,0,2$		

As before, the arrows and labels indicate from which elements each number was derived. The shaded cells and arrows correspond to traceback; it starts at $F(3,2,2) = -1$ and goes through the shaded cells until we reach $F(0,0,0)$. The traceback produces the following three paths:

$$F(3,2,2) \rightarrow F(2,1,1) \rightarrow F(1,0,0) \rightarrow F(0,0,0)$$

$$F(3,2,2) \rightarrow F(2,1,1) \rightarrow F(1,1,1) \rightarrow F(0,0,0)$$

$$F(3,2,2) \rightarrow F(2,2,2) \rightarrow F(1,1,1) \rightarrow F(0,0,0)$$

They respectively give rise to the following three optimal alignments with score -1:

$$\begin{array}{llllll} x: & A & A & T & C \\ y: & - & G & T & C \\ z: & - & A & A & G \end{array}$$

$$\begin{array}{llllll} x: & A & A & T & C \\ y: & G & - & T & C \\ z: & A & - & A & G \end{array}$$

$$\begin{array}{llllll} x: & A & A & T & C \\ y: & G & T & - & C \\ z: & A & A & - & G \end{array}$$

Because of the memory and time complexity, the above algorithm in practice cannot be applied to align a large number of sequences. Therefore, alternative algorithms (mainly heuristic) have been developed. Below we briefly mention some of them.

3.5 Star Alignment

Suppose we are given the following five DNA sequences:

x_1 : A T T G C C A T T
x_2 : A T G G C C A T T
x_3 : A T C C A A T T T T
x_4: A T C T T C T T
x_5 : A C T G A C C

We assume the same scoring scheme for pairwise alignments as in the global alignment section and consider the corresponding SP-scoring scheme with linear gap model. We calculate all pairwise optimal scores (that is, the scores found by the global pairwise alignment algorithm described), write them in the matrix below, and find the sum in each row:

	x_1	x_2	x_3	x_4	x_5	Total Score
x_1		7	-2	0	-3	2
x_2	7		-2	0	-4	1
x_3	-2	-2		0	-7	-11
x_4	0	0	0		-3	-3
x_5	-3	-4	-7	-3		-17

Of all the sequences, x^1 has the best total score (equal to 2) and is selected to be at the center of the future star. The optimal alignments between x^1 and each of the other sequences found by the global alignment algorithm from Sect. 4.1 are as follows:

x_1 : A T T G C C A T T
x_2 : A T G G C C A T T
x_1 : A T T G C C A T T _ _
x_3 : A T C - C A A T T T T
x_1 : A T T G C C A T T
x_4 : A T C T T C _ T T
x_1 : A T T G C C A T T
x_5 : A C T G A C C _ _

We now merge the above alignments using the" once a gap– always a gap" principle. We start with x^1 and x^2:

x^1 : A T T G C C A T T
x^2 : A T G G C C A T T

and add x^3, but since x^3 is longer than x^1 and x^2, we add gaps at the ends of x^1 and x^2:

x^1 : A T T G C C A T T _ _

x^2 : A T G G C C A T T _ _
x^3 : A T C _C A A T T T T

3. CONCLUSION

In conclusion, this study demonstrates the effectiveness of dynamic programming algorithms in analyzing DNA sequences obtained from different ciliates during meiotic reproduction. By using a combination of global and local alignment algorithms, affine gap and star alignment algorithms, and multiple alignment algorithms, we were able to identify conserved regions and patterns in the DNA sequences, and construct DNA sequences that reflected the evolutionary relationships among the ciliates. Our findings have important implications for our understanding of the evolution and diversity of ciliates and other organisms, and highlight the utility of dynamic programming algorithms in analyzing complex biological data. Future research in this area could build on our findings by utilizing these algorithms to analyze DNA sequences from a wider range of organisms, and by incorporating additional data sources to further refine our understanding of the evolutionary relationships among these organisms.

References

Aguileta, G., de Vienne, D. M., Ross, O. N., Hood, M. E., Giraud, T., Petit, E., and Gabald´on, T. (2014). High variability of mitochondrial gene order among fungi. Genome Biol. Evol., 6(2):451–465.

Archibald, J. M., Simpson, A. G. B., and Slamovits, C. H. (2017). Handbook of the protists. page 691. Springer International Publishing, 2 edition.

Beermann, S. (1977). The diminution of heterochromatic chromosomal segments in cyclops (crustacea, copepoda). Chromosoma, 60(4):297–344.

Biller, P., Gu´ eguen, L., Knibbe, C., and Tannier, ´ E. (2016). Breaking good: accounting for fragility of genomic regions in rearrangement distance estimation. Genome Biol. Evol., 8(5):1427–1439.

Dobzhansky, T. (1933). On the sterility of the interracial hybrids in drosophila pseudoobscura. Proc. Natl Acad. Sci., 19(4):397–403.

Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D., and Rozenberg, G. (2004). Computation in Living Cells—Gene Assembly in Ciliates. Springer Verlag, Berlin.

Foissner, W. and Hawksworth, D. (2009). Protist Diversity and Geographical Distribution, volume 8 of Topics in Biodiversity and Conservation. Springer Netherlands.

Gerbi, S. A. (1986). Unusual chromosome movements in sciarid flies. Results Probl. Cell Differ., 13:71104.

Gotoh, O. (1982). An improved algorithm for matching biological sequences. Journal of molecular biology, 162(4):705–708.

Lang, B. F., Jakubkova, M., Hegedusova, E., Daoud, R., Forget, L., Brejova, B., Vinar, T., Kosa, P., Fricova, D., Nebohacova, M., et al. (2014). Massive programmed translational jumping in mitochondria. Proc. Natl Acad. Sci., 111(16):5926–5931.

Lynn, D. (2008). The Ciliated Protozoa. Springer, 3rd edition.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of molecular biology, 48(3):443–453.

Nielsen, T. G. and Kiørboe, T. (1994). Regulation of zooplankton biomass and production in a temperate, coastal ecosystem. 2. ciliates. Limnology and Oceanography, 39(3):508–519.

Orr, H. A. (1996). Dobzhansky, bateson, and the genetics of speciation. Genetics, 144(4):1331–1335.

Pevzner, P. and Tesler, G. (2003). Genome rearrangements in mammalian evolution: lessons from human and mouse genomes. Genome Res., 13(1):37–45.

Pevzner, P. A., Tang, H., and Waterman, M. S. (2001). An eulerian path approach to dna fragment assembly. Proc. Natl Acad. Sci., 98(17):9748–9753.

Prescott, D. M. (1994). The dna of ciliated protozoa. Microbiol. Rev., 58(2):233–267.

Rotterov´a, J., Edgcomb, V. P., ˇ Cepiˇ cka, I., and Beinart, R. (2022). Anaerobic ciliates as a model group for studying symbioses in oxygen-depleted environments. The Journal of Eukaryotic Microbiology, 69(5):e12912.

Smith, J. J., Baker, C., Eichler, E. E., and Amemiya, C. T. (2012). Genetic consequences of programmed genome rearrangement. Curr. Biol., 22(16):1524–1529.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. Journal of molecular biology, 147(1):195–197.

Stephens, P. J., Greenman, C. D., Fu, B., Yang, F., Bignell, G. R., Mudie, L. J., Pleasance, E. D., Lau, K. W., Beare, D., Stebbings, L. A., et al. (2011). Massive genomic rearrangement acquired in a single catastrophic event during cancer development. Cell, 144(1):27–40.

Yancopoulos, S., Attie, O., and Friedberg, R. (2005). Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics, 21(16):3340–3346