02-710 Computational Genomics Project Report

Bioinformatics Tool: Pairwise Sequence Alignment Visualizer

Yunshuo Chou (yunshuoc), Jialan Ma (jialanm), Tian Liu (tianl2)

Introduction

In the rapidly growing field of bioinformatics, comparing and analyzing biological sequences has become a fundamental task, essential to understanding the structure, function, and evolutionary relationships between species. Pairwise sequence alignment, which involves aligning two sequences to identify regions of similarity, has been a cornerstone technique in this field. The information obtained from these alignments can provide insights into conserved regions, functional domains, or regions under selective pressure, contributing to the development of novel therapeutic targets, diagnostic markers, and evolutionary studies.

Despite the importance of pairwise sequence alignment, visualizing and interpreting the results can be challenging, particularly for large or complex sequences. Various software and tools have been developed to address this issue. However, many existing tools have limitations, such as limited platform compatibility, or the need for local installations, which can hinder their widespread adoption and ease of use.

Our Pairwise Sequence Alignment Visualizer (Align-Viz) is a bioinformatics tool designed to facilitate the visualization, interpretation, and analysis of sequence alignment results. It aims to provide researchers, educators, and students with an intuitive, user-friendly, and feature-rich platform, enabling them to gain insights into sequence relationships without the need for extensive bioinformatics expertise or external software. Furthermore, the Align-Viz was designed as a web-based application, ensuring its accessibility and eliminating the need for local installations or specialized hardware.

Methods

Our Pairwise Sequence Alignment Visualizer (Align-Viz) is implemented using Next.js, a popular React framework for building web applications. The application is designed with a modular structure, allowing easy incorporation of additional alignment algorithms and visualization techniques in the future. For the underlying pairwise sequence alignment algorithm, we employed dynamic programming-based algorithms to search for the optimal alignment of two sequences. The primary components of the Align-Viz are as follows:

Algorithm

We employ two widely used pairwise sequence alignment algorithms, Needleman-Wunsch for global alignment and Smith-Waterman for local alignment. These algorithms are implemented in separate modules (GlobalAlignment and LocalAlignment) and use dynamic programming techniques to compute the optimal alignment between two unlimited sequences, considering user-defined scoring schemes for matches, mismatches, and gaps.

The Needleman-Wunsch algorithm and the Smith-Waterman algorithm are classic dynamic programming algorithms used for global sequence alignment and local sequence alignment, respectively. The algorithms are designed to find the optimal global and local alignment of two sequences, respectively, by considering all possible alignments and scoring them based on a scoring scheme, which typically includes match, mismatch, and gap scores. The optimal alignments are computed through identifying the optimal scores and backtracking from the optimal scores to find the matching patterns.

While the two algorithms share some similarities in their dynamic programming approach, they differ in their initialization, matrix filling rules, and traceback procedures. The Smith-Waterman algorithm enables more precise sequence alignment without having to align the ends of related sequences that may be highly different. That is, the algorithm allows for free gaps in the beginning and the end, and score resetting at each step. In contrast, the Needleman-Wunsch algorithm aims to achieve the best global alignment across the entire input, penalizing gaps and disabling score resetting.

Visualization and User Interface

The user interface is built using React components and custom CSS styles. The main components include:
TopBar: A navigation bar that allows users to input sequences, select alignment mode (Global or Local), and specify scoring parameters. Additionally, users can initiate the alignment process and reset the visualization.
TableGrid: Displays the dynamic programming matrix, with rows and columns corresponding to the input sequences. The matrix is updated in real-time during the alignment process, visualizing the algorithm's progress.
GlobalAlignedStr and **LocalAlignedStr**: These components display the final aligned sequences for global and local alignment, respectively.
Upon receiving user input, the application triggers the selected alignment algorithm, updates the dynamic programming matrix, and visualizes the optimal alignment path. AnimateNW and animateSW functions are responsible for animating the Needleman-Wunsch and Smith-Waterman algorithms, respectively. These functions update the matrix cell status (visited, current, path) and employ a delay between state updates to visualize the algorithm's progress.

The Align-Viz is deployed on the Vercel platform, making it accessible as a web-based application without the need for local installation. Check out the deployed demo: https://align-viz.vercel.app/

Implementation details

Global Alignment

The needlemanWunsch function in the GlobalAlignment.ts file performs global sequence alignment using the Needleman-Wunsch algorithm. The inputs to the function are two sequences (seq1, seq2), scores for matches (matchScore), mismatches (mismatchScore), and gaps (gapScore). The function returns an object containing the dynamic programming matrix, the optimal alignment, and the global path.

The matrix is initialized with zeros, and the first row and column are filled with gap scores. Then, the matrix is filled by iterating through each cell and calculating the optimal score using the match, mismatch, and gap scores. Each entries in the matrix is calculated by:
$O(i, j) = max(O(i - 1, j - 1) + s(x_i, y_j), O(i - 1, j) + g, O(i, j - 1) + g),$
In particular, the $s(x_i, y_j)$ denotes the match/mismatch score for the current i,j pair, while g is the predefined gap score.

The direction of each cell is also determined during this process. After filling the matrix, the optimal alignment is found using backtracking, starting from the bottom-right corner to the top-left corner of the matrix.

Local Alignment

The SmithWaterman function in the LocalAlignment.ts file performs local sequence alignment using the Smith-Waterman algorithm. The inputs to the function are two sequences (seq1, seq2), scores for matches (matchScore), mismatches (mismatchScore), and gaps (gapScore). The function returns an object containing the score matrix, the traceback alignments, the local paths, and the maximum score in the matrix.

Similar to the Needleman-Wunsch algorithm, the Smith-Waterman algorithm initializes a scoring matrix with zeros. However, the Smith-Waterman algorithm allows for negative values to be replaced with zeros when filling the matrix. By doing this, we effectively discard any negative-scoring alignments and focus on the highest-scoring local alignment within the sequences. Each entries in the matrix is calculated by:
$O(i, j) = max(O(i - 1, j - 1) + s(x_i, y_j), O(i - 1, j) + g, O(i, j - 1) + g, 0),$
In particular, the $s(x_i, y_j)$ denotes the match/mismatch score for the current i,j pair, while g is the predefined gap score.

After filling the matrix, local maxima are identified, and a traceback is performed for each local maximum. The traceback function (traceback) is a recursive function that backtracks through the matrix, constructing the optimal local alignments.

The local alignment results include the score matrix, the traceback alignments, the local paths, and the maximum score in the matrix. These values are returned to the main component for animation.

React components

Cell.tsx:

The Cell component is responsible for rendering individual cells of the dynamic programming table. It takes several properties as input, such as cellSize, content, cellStatus, and bgColor. The component uses conditional rendering to determine the appropriate background color and class name for the cell, based on its cellStatus. The cellStatus property is essential to the animation to work, as it determines the background color for each cell base on the current state. The cellStyle object is then created with the computed background color, cellSize, and other style attributes. Finally, the component returns a div element with the applied styles and content.

GlobalAlignedString.tsx:

The GlobalAlignedString component displays the global alignment of two input strings. It accepts the input strings as properties and renders them in a flex container with two grid elements, one for each string. The grid elements use gridTemplateColumns to distribute the characters of the input strings evenly across the available width. This is crucial for the layout to stay consistent for long input sequences. The component then maps the characters of each string to individual span elements and renders them within their respective grid containers.

LocalAlignedString.tsx:

The LocalAlignedString component is responsible for rendering the local alignments of pairs of input strings. This components is independent from the above GlobalAlignedString component because there may be multiple local alignment result strings in the input sequences so it requires a different design. It takes an array of string pairs as input and creates a flex container with a fixed height and overflow-y set to auto. For each pair of input strings, the component calculates the maximum length of the strings and renders them in separate grid containers with gridTemplateColumns adjusted according to the maxLength. The characters of each string are mapped to span elements and rendered within their respective grid containers. Border styles are applied to separate the alignments.

TableGrid.tsx:

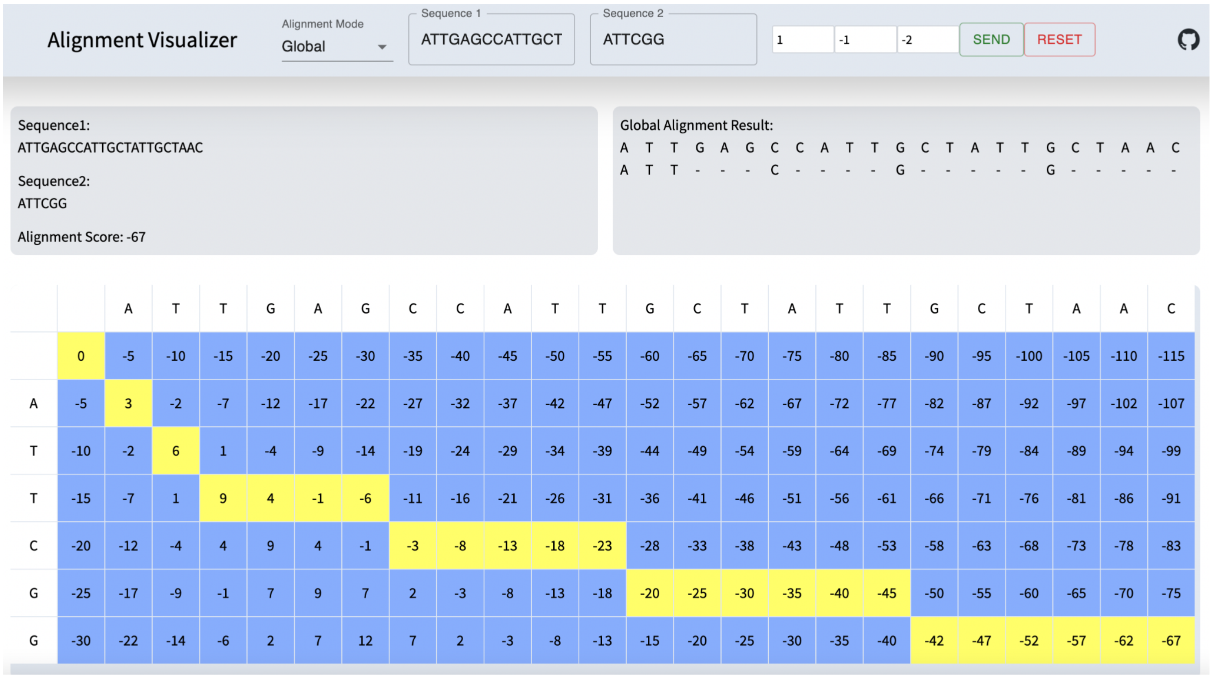
The TableGrid component is responsible for rendering the dynamic programming table for the sequence alignment algorithm. It receives a displayMatrix, seq1, and seq2 as properties. The component calculates the number of rows and columns based on the lengths of the input sequences, and creates a tableStyle object to define the grid dimensions and gap. It then iterates through the rows and columns, populating the table with Cell components. The displayMatrix is used to determine the content and cellStatus of the Cell components for each cell in the dynamic programming table.

TopBar.tsx:
The TopBar component serves as the main interface for user input, allowing users to enter sequences, alignment mode, and scoring parameters. It uses Material-UI components, such as TextField, Button, Select, MenuItem, InputLabel, and FormControl, to create a responsive and interactive user interface. The TopBar component also includes functions to handle user input, such as handleSend and handleReset. These functions are called when the "Send" and "Reset" buttons are clicked, respectively. The component also provides a link to the GitHub repository of the project, allowing users to access the source code directly.
In summary, the Sequence Alignment Visualizer consists of several React components that work together to provide an interactive and visual representation of sequence alignment algorithms. Users can input sequences, choose between global and local alignment modes, and customize scoring parameters to observe how the dynamic programming table is populated and how the alignments are determined.

Results and Conclusions

Our tool Align-Viz provides an interactive and user-friendly platform to visualize the process of pairwise sequence alignment. It implements the well-established Needleman-Wunsch and Smith-Waterman algorithms for global and local alignments, respectively. This tool enables researchers and students to gain a better understanding of these algorithms and their applications in biological sequence analysis.

The following figures are snapshots of the Align-Viz interface, displaying the input sequences, alignment mode, scoring parameters, and the resulting alignment matrix. Also, we did a comparison of global (Needleman-Wunsch) and local (Smith-Waterman) alignments for a pair of input sequences, showcasing the differences in the resulting alignments:



Alignment Mode: Global
Sequence 1: ATTGAGCCATTGCT
Sequence 2: ATTCGG
1 -1 -2

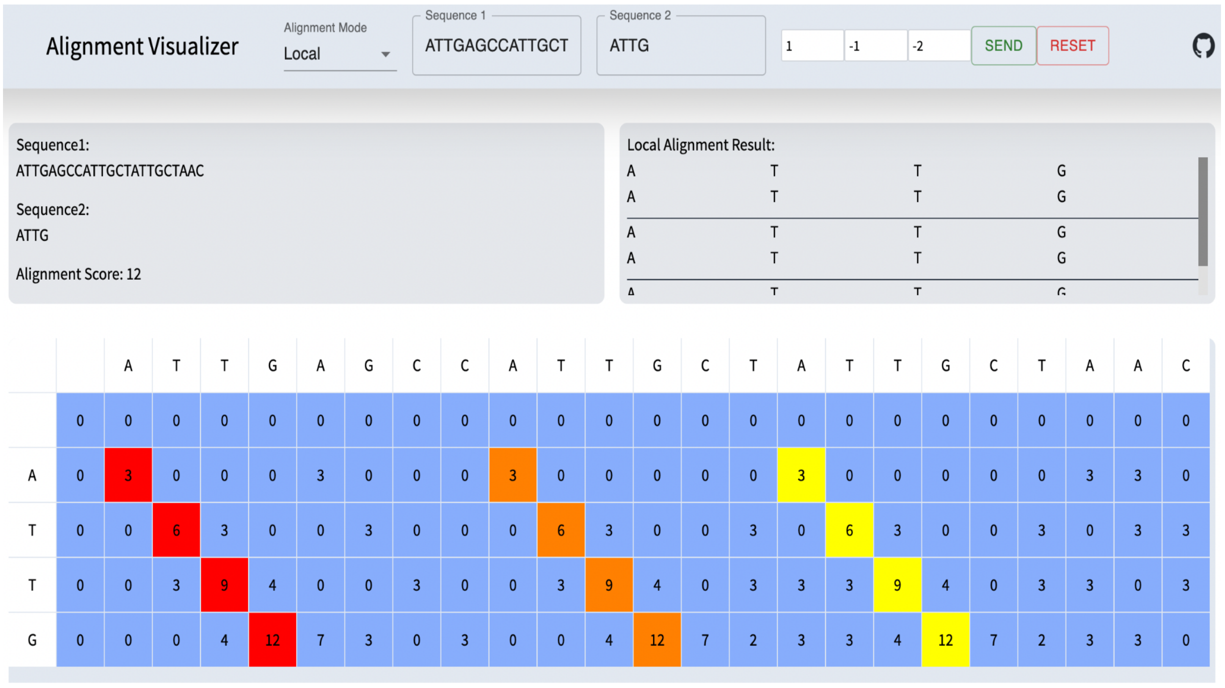
Sequence1:
ATTGAGCCATTGCTATTGCTAAC

Sequence2:
ATTCGG

Alignment Score: -67

Global Alignment Result:
```
A T T G A G C C A T T G C T A T T G C T A A C
A T T - - - C - - - - G - - - - - G - - - - -
```

		A	T	T	G	A	G	C	C	A	T	T	G	C	T	A	T	T	G	C	T	A	A	C
	0	-5	-10	-15	-20	-25	-30	-35	-40	-45	-50	-55	-60	-65	-70	-75	-80	-85	-90	-95	-100	-105	-110	-115
A	-5	3	-2	-7	-12	-17	-22	-27	-32	-37	-42	-47	-52	-57	-62	-67	-72	-77	-82	-87	-92	-97	-102	-107
T	-10	-2	6	1	-4	-9	-14	-19	-24	-29	-34	-39	-44	-49	-54	-59	-64	-69	-74	-79	-84	-89	-94	-99
T	-15	-7	1	9	4	-1	-6	-11	-16	-21	-26	-31	-36	-41	-46	-51	-56	-61	-66	-71	-76	-81	-86	-91
C	-20	-12	-4	4	9	4	-1	-3	-8	-13	-18	-23	-28	-33	-38	-43	-48	-53	-58	-63	-68	-73	-78	-83
G	-25	-17	-9	-1	7	9	7	2	-3	-8	-13	-18	-20	-25	-30	-35	-40	-45	-50	-55	-60	-65	-70	-75
G	-30	-22	-14	-6	2	7	12	7	2	-3	-8	-13	-15	-20	-25	-30	-35	-40	-42	-47	-52	-57	-62	-67



Alignment Mode: Local
Sequence 1: ATTGAGCCATTGCT
Sequence 2: ATTG
1 -1 -2

Sequence1:
ATTGAGCCATTGCTATTGCTAAC

Sequence2:
ATTG

Alignment Score: 12

Local Alignment Result:

```
A    T    T    G
A    T    T    G
-----------------
A    T    T    G
A    T    T    G
-----------------
A    T    T    G
```

		A	T	T	G	A	G	C	C	A	T	T	G	C	T	A	T	T	G	C	T	A	A	C
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	3	0	0	0	3	0	0	0	3	0	0	0	0	0	3	0	0	0	0	0	3	3	0
T	0	0	6	3	0	0	3	0	0	0	6	3	0	0	3	0	6	3	0	0	3	0	3	3
T	0	0	3	9	4	0	0	3	0	0	3	9	4	0	3	3	3	9	4	0	3	3	0	3
G	0	0	0	4	12	7	3	0	3	0	0	4	12	7	2	3	3	4	12	7	2	3	3	0

In summary, the Align-Viz effectively demonstrates the process of pairwise sequence alignment using Needleman-Wunsch and Smith-Waterman algorithms. Users can visualize how the algorithms work, enabling a deeper understanding of their inner workings and applications. This tool facilitates the analysis of sequence alignments and their potential biological implications.

There are some limitations to our current Align-Viz tool. Firstly, the implementation is focused on pairwise alignments and does not support multiple sequence alignments. Performance limitations may also arise when aligning very long sequences due to the computational complexity of the algorithms. Furthermore, the visualizer currently lacks advanced features, such as the incorporation of different substitution matrices or custom scoring schemes. Presently, the visualization tool only supports webpages, but it should be made responsive to accommodate users who wish to access the application on tablets or phones for added convenience.

To address these limitations, we have several future directions planned. We aim to expand the tool to support multiple sequence alignments, utilizing algorithms such as ClustalW or including heuristic methods to optimize the performance. Additionally, we will work to enhance performance for long sequence alignments by implementing more efficient algorithms or parallel computing techniques. Moreover, we intend to implement features that enable importing and exporting alignments in standard formats like FASTA and Clustal. This improvement will eliminate the need for users to manually input sequences each time they use the tool.

Code Availability

The code is available on github: https://github.com/ChouYunShuo/Align-Viz. The current version of the toy demonstration is deployed using Vercel.

References

https://nextjs.org/docs
https://en.wikipedia.org/wiki/Sequence_alignment
https://mui.com/material-ui/guides/typescript/
https://tailwindcss.com/docs/theme
https://github.com/ChouYunShuo/Align-Viz
https://medium.com/analytics-vidhya/sequence-alignment-and-the-needleman-wunsch-algorithm-710c7b1a23a4