SMITH-WATERMAN ALGORITHM

Sai Ram Gurram
CSE 708 (Dr. Russ Miller)
Table of Contents

- Introduction
- Applications
- Smith Waterman Algorithm with an example
- Need for Parallelization
- Results
Smith-Waterman Algorithm

- The Smith-Waterman algorithm is a well-known method used in bioinformatics for local sequence alignment; that is, for aligning subsequences of proteins or nucleic acids.
- The Smith-Waterman algorithm is a dynamic programming algorithm used for local sequence alignment.
- It compares segments of all possible lengths and optimizes the measure of similarity.
- The goal is to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences.
SMITH WATERMAN

NEEDLEMAN–WUNSCH
Algorithm

- Consider the maximum value from 4 cases
- Apply Gap penalty to top and left
- Match/Mismatch score to the diagonal value
- Set the score to zero if it negative

Match/Mismatch: +3/-3
Gap penalty: -2

\[ H_{ij} = \max \left\{ \begin{array}{l} H_{i-1,j-1} + s(a_i, b_j), \\ \max_{k \geq 1} \{H_{i-k,j} - W_k\}, \\ \max_{l \geq 1} \{H_{i,j-l} - W_l\}, \\ 0 \end{array} \right. \quad (1 \leq i \leq n, 1 \leq j \leq m) \]

where

- \( H_{i-1,j-1} + s(a_i, b_j) \) is the score of aligning \( a_i \) and \( b_j \),
- \( H_{i-k,j} - W_k \) is the score if \( a_i \) is at the end of a gap of length \( k \),
- \( H_{i,j-l} - W_l \) is the score if \( b_j \) is at the end of a gap of length \( l \),
- 0 means there is no similarity up to \( a_i \) and \( b_j \).
The Alignment result is:

G T T – A C
G T T G A C
Need for Parallel

- Sequence alignment is a highly time-consuming task as it takes quadratic time which is \( O(mn) \), \( m,n \) are the lengths of the protein sequences
- Let’s take an example of (COVID-19), scientists identified its common features by aligning it against other viruses
Parallel Implementation

• Each cell in the computation matrix depends on the calculations of three other cells, as indicated by red arrows in the document.

• Diagonal elements doesn’t depend on the other diagonal elements so they can be computed parallelly
Parallel Implementation

Figure 2. Cases calculable at the same time $T_i$.

Figure 3. Linear representation of the parallelizable boxes.
Results

For dataset $N = 1000$
For $N = 2000$

**Processing Time vs. Number of Processors**

**Speedup vs. Number of Processors**
For $N = 5000$
For N = 10000
References


• https://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm

• Parallelizing the Smith-Waterman Algorithm using OpenSHMEM and MPI-3 One-Sided Interfaces - Matthew Baker, Aaron Welch, Manjunath Gorentla Venkata.
Thank You