# Dynamic Programming & Smith-Waterman algorithm

## Seminar: Classical Papers in Bioinformatics

Yvonne Herrmann

May 3rd, 2010

# Overview

# Dynamic Programming
## Introduction





### Definition

- Dynamic Programming is a method of solving problems by breaking them down into simpler steps
- problem need to contain overlapping subproblems and should have an optimal substructure
- method is used for mathematical optimization and computer programming

# Dynamic Programming
## Introduction





### Definition

- Dynamic Programming is a method of solving problems by breaking them down into simpler steps
- problem need to contain **overlapping** subproblems and should have an optimal substructure
- method is used for mathematical optimization and computer programming

# Dynamic Programming
## Introduction



### Divide&Conquer

- Divide&Conquer is used when all subproblems are independent.
  - calculate partitions and combine the solutions to solve the entire problem.

vs.

### Dynamic Programming

- Dynamic Programming is used when subproblems are dependent
  - there are no partitions, since the subproblems overlap.

# Dynamic Programming
## Introduction





### Definition

- Dynamic Programming is a method of solving problems by breaking them down into simpler steps
- problem need to contain overlapping subproblems and should have an **optimal substructure**
- method is used for mathematical optimization and computer programming

## The Principle of Optimality

*"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."* [a]

[a] Bellman, R.E. 1957. Dynamic Programming, Chap.III.3., Princeton University Press

## shortest path

- shortest way by car to get from Bielefeld to Cologne
- have to pass through Hamm(Westf) and Dortmund
- shortest route from Hamm(Westf) to Cologne, needs to go through Dortmund

$\Rightarrow$ *The second problem is inside the first one.*

## Dynamic Programming is used by...

- Floyd-Warshall algorithm (shortest path algorithm)
- Needleman-Wunsch algorithm
- **Smith-Waterman algorithm**
- Bellman-Ford algorithm, etc.

# Overview

## Why compare sequences?

Quantify the similarity or dissimilarity between two or more sequences and find out where they are similar or different.

# Sequence comparison
## Why compare sequences?



The analysis of this can help to determint:

- if genes from two different organism are related
- if similar nucleotide sequences lead to similar protein structures
- which species is likely more related to another one
- what kind of development happened in the evolution? (Mutations, insertions and deletions of gens or more specific in the aminoacid sequence itself)

# Alignments
## How to compare sequences?



**sequence alignment**

Method of arranging the sequences of DNA, RNA or aminoacids of proteins to find regions of similarity which might be a consequence of functional, structural or evolutionary relationships between the sequences.

# Alignments
## How to compare sequences?



### Conditions a alignment has to fulfill

- all symbols have to be in the same order they appear in the given sequences
- a symbol can be aligned with a blank ('-')
- two blanks cannot be aligned

# Alignments
## How to compare sequences?



### Example

sequence s and t are given:

s: A C T G A A C T G

t: A T G G A C C T G

a possible alignment is:

A C T - G A - A C T G

A - T G G A C - C T G

# Local vs. global alignment
## What's the difference?



### global alignment

The sequences must be aligned from start to end.

### local alignment

Local alignments identify regions of high similarity within sequences.

# Local vs. global alignment
## What's the difference?



### global alignment

The sequences must be aligned from start to end.

### local alignment

Local alignments identify regions of high similarity within sequences which are often widely different overall.

Smith-Waterman algorithm calculates the optimal local alignment!

# Overview



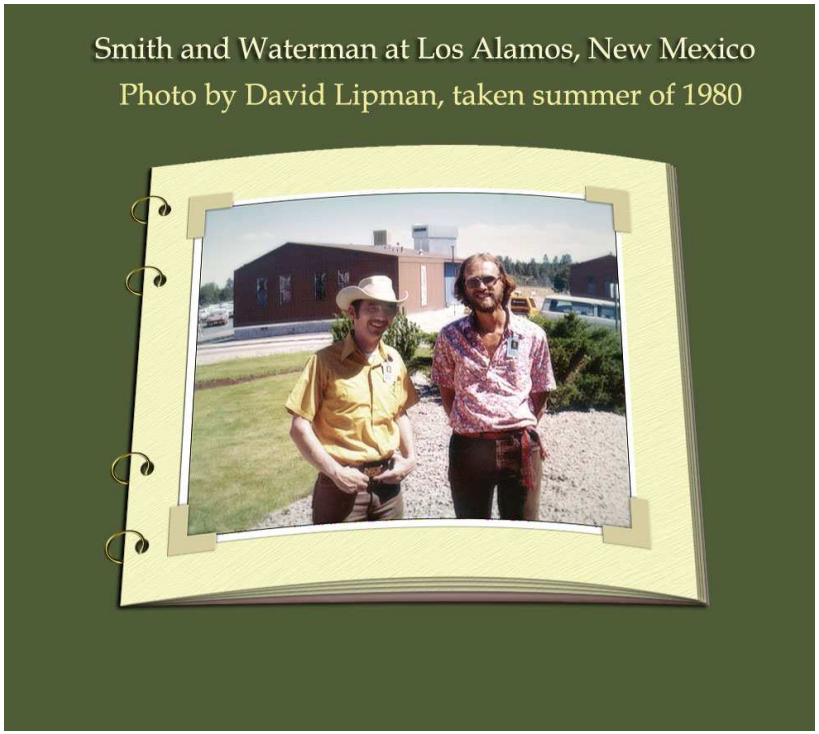


1. Dynamic Programming
2. Sequence comparison
   - Intentions
   - Alignments
3. Smith-Waterman algorithm

# Smith-Waterman algorithm
## A little history



- algorithm was proposed in 1981 by Temple F. Smith and Michael S. Waterman
- algorithm uses dynamic programming and is a variation of the Needleman-Wunsch algorithm



# Smith-Waterman algorithm
## What's the goal of this algorithm?



- Smith-Waterman algorithm calculates the local alignment of two given sequences
- used to identify similar DNA, RNA and protein segments
- alignments of any possible length starting and ending at any position in the two sequences are compared to obtain the optimal local alignment

# Smith-Waterman algorithm
## What's the goal of this algorithm?



- it guarantees to find the optimal local alignment considering the given scoring system.
- scoring system includes a substitution matrix and a gap-scoring scheme.
- scores consider matches, mismatches, substitutions or insertions/deletions
- main difference to the Needleman-Wunsch algorithm is: negative scores are set to zero

## Starting conditions

- two molecular sequences A=$a_1 a_2 \ldots a_n$ and B=$b_1 b_2 \ldots b_m$.
- scoring theme

## course of events

- first: setting up matrix H
  $H_{k0} = H_{0l} = 0$ ($for\ 0 \leq k \leq n\ and\ 0 \leq l \leq m$)
- next: calculate score for each cell
- last: backtrace the path to obtain optimal alignment

## How to calculate the score for each cell?

Individual pair-wise comparisons between the characters as:

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ max_k\{ H_{i-k,j} - W_k\}, \\ max_l\{ H_{i,j-l} - W_l\}, \\ 0. \end{cases}$$

k = deletion of length k

l = deletion of length l

$W_k$ *and* $W_l$ *is the gap − cost function*

## backtracing

During the filling of matrix H you have to use backpointers to reconstruct from which cell you came.
Then when you found the highest score in the matrix H you can backtrace the path and obtain the optimal alignment.

*caption of backpointers:*

| | |
|---|---|
| $\leftarrow$ | *Deletion* |
| $\uparrow$ | *Insertion* |
| $\nwarrow$ | *Substitution* |

## Example

sequence A and B are given:

A: A G C T T and B: A G A C T

scoring theme:

$$match = +1$$

$$mismatch = -\frac{1}{3}$$

$$W_k = 1 + \frac{1}{3} * k$$

## Example

sequence A and B are given:
A: A G C T T and B: A G A C T

| | θ | A | G | C | T | T |
|---|---|---|---|---|---|---|
| θ | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| A | 0,0 | 1,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| G | 0,0 | 0,0 | 2,0 | 0,7 | 0,0 | 0,0 |
| A | 0,0 | 1,0 | 0,7 | 1,7 | 0,3 | 0,0 |
| C | 0,0 | 0,0 | 0,0 | 1,7 | 0,3 | 0,0 |
| T | 0,0 | 0,0 | 0,0 | 0,3 | 2,7 | 2,3 |
| | | | | | | |

Figure: Filled matrix H

## Example

optimal local alignment:

A G A C T

A G - C T

| | θ | A | G | C | T | T |
|---|---|---|---|---|---|---|
| θ | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| A | 0,0 | 1,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| G | 0,0 | 0,0 | 2,0 | 0,7 | 0,0 | 0,0 |
| A | 0,0 | 1,0 | 0,7 | 1,7 | 0,3 | 0,0 |
| C | 0,0 | 0,0 | 0,0 | 1,7 | 0,3 | 0,0 |
| T | 0,0 | 0,0 | 0,0 | 0,3 | 2,7 | 2,3 |
| | | | | | | |

Figure: Filled matrix H and backtracing path

# Smith-Waterman algorithm
## Smith-Waterman - Example 2



- best optimal local alignment can be anywhere in the sequences
  $\rightarrow$ *Find highest score in matrix H as backtracing start point*

| | θ | C | A | G | C | C | U | C | G | C | U | U | A | G |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| θ | 0,0 | 0,0 | 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 | 0,0 | 0,0 | 1,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| A | 0,0 | 0,0 | 1,0 | 0,7 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,7 |
| U | 0,0 | 0,0 | 0,0 | 0,7 | 0,3 | 0,0 | 1,0 | 0,0 | 0,0 | 0,0 | 1,0 | 1,0 | 1,0 | 0,7 |
| G | 0,0 | 0,0 | 0,0 | 1,0 | 0,3 | 0,0 | 0,0 | 0,7 | 1,0 | 0,0 | 0,0 | 0,7 | 0,7 | 1,0 |
| C | 0,0 | 1,0 | 0,0 | 0,0 | 2,0 | 1,3 | 0,3 | 1,0 | 0,3 | 2,0 | 0,7 | 0,3 | 0,3 | 0,3 |
| C | 0,0 | 1,0 | 0,7 | 0,0 | 1,0 | 3,0 | 1,7 | 1,3 | 1,0 | 1,3 | 1,7 | 0,3 | 0,3 | 0,0 |
| A | 0,0 | 0,0 | 2,0 | 0,7 | 0,3 | 1,7 | 2,7 | 1,3 | 1,0 | 0,7 | 1,0 | 1,3 | 1,3 | 0,0 |
| U | 0,0 | 0,0 | 0,7 | 1,7 | 0,3 | 1,3 | 2,7 | 2,3 | 1,0 | 0,7 | 1,7 | 2,0 | 1,0 | 1,0 |
| U | 0,0 | 0,0 | 0,3 | 0,3 | 1,3 | 1,0 | 2,3 | 2,3 | 2,0 | 0,7 | 1,7 | 2,7 | 1,7 | 1,0 |
| G | 0,0 | 0,0 | 0,0 | 1,3 | 0,0 | 1,0 | 1,0 | 2,0 | 3,3 | 2,0 | 1,7 | 1,3 | 2,3 | 2,7 |
| A | 0,0 | 0,0 | 1,0 | 0,0 | 1,0 | 0,3 | 0,7 | 0,7 | 2,0 | 3,0 | 1,7 | 1,3 | 2,3 | 2,0 |
| C | 0,0 | 1,0 | 0,0 | 0,7 | 1,0 | 2,0 | 0,7 | 1,7 | 1,7 | 3,0 | 2,7 | 1,3 | 1,0 | 2,0 |
| G | 0,0 | 0,0 | 0,7 | 1,0 | 0,3 | 0,7 | 1,7 | 0,3 | 2,7 | 1,7 | 2,7 | 2,3 | 1,0 | 2,0 |
| G | 0,0 | 0,0 | 0,0 | 1,7 | 0,3 | 0,3 | 0,3 | 1,3 | 1,3 | 2,3 | 1,3 | 2,3 | 2,0 | 2,0 |

Figure: Example from the original paper

# Smith-Waterman algorithm
## Smith-Waterman - Example 2



optimal local alignment:

G C A U U G

G C - U C G

| | θ | C | A | G | C | C | U | C | G | C | U | U | A | G |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| θ | 0,0 | 0,0 | 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 | 0,0 | 0,0 | 1,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| A | 0,0 | 0,0 | 1,0 | 0,7 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,7 |
| U | 0,0 | 0,0 | 0,0 | 0,7 | 0,3 | 0,0 | 1,0 | 0,0 | 0,0 | 0,0 | 1,0 | 1,0 | 1,0 | 0,7 |
| G | 0,0 | 0,0 | 0,0 | 1,0 | 0,3 | 0,0 | 0,0 | 0,7 | 1,0 | 0,0 | 0,0 | 0,7 | 0,7 | 1,0 |
| C | 0,0 | 1,0 | 0,0 | 0,0 | 2,0 | 1,3 | 0,3 | 1,0 | 0,3 | 2,0 | 0,7 | 0,3 | 0,3 | 0,3 |
| C | 0,0 | 1,0 | 0,7 | 0,0 | 1,0 | 3,0 | 1,7 | 1,3 | 1,0 | 1,3 | 1,7 | 0,3 | 0,3 | 0,0 |
| A | 0,0 | 0,0 | 2,0 | 0,7 | 0,3 | 1,7 | 2,7 | 1,3 | 1,0 | 0,7 | 1,0 | 1,3 | 1,3 | 0,0 |
| U | 0,0 | 0,0 | 0,7 | 1,7 | 0,3 | 1,3 | 2,7 | 2,3 | 1,0 | 0,7 | 1,7 | 2,0 | 1,0 | 1,0 |
| U | 0,0 | 0,0 | 0,3 | 0,3 | 1,3 | 1,0 | 2,3 | 2,3 | 2,0 | 0,7 | 1,7 | 2,7 | 1,7 | 1,0 |
| G | 0,0 | 0,0 | 0,0 | 1,3 | 0,0 | 1,0 | 1,0 | 2,0 | 3,3 | 2,0 | 1,7 | 1,3 | 2,3 | 2,7 |
| A | 0,0 | 0,0 | 1,0 | 0,0 | 1,0 | 0,3 | 0,7 | 0,7 | 2,0 | 3,0 | 1,7 | 1,3 | 2,3 | 2,0 |
| C | 0,0 | 1,0 | 0,0 | 0,7 | 1,0 | 2,0 | 0,7 | 1,7 | 1,7 | 3,0 | 2,7 | 1,3 | 1,0 | 2,0 |
| G | 0,0 | 0,0 | 0,7 | 1,0 | 0,3 | 0,7 | 1,7 | 0,3 | 2,7 | 1,7 | 2,7 | 2,3 | 1,0 | 2,0 |
| G | 0,0 | 0,0 | 0,0 | 1,7 | 0,3 | 0,3 | 0,3 | 1,3 | 1,3 | 2,3 | 1,3 | 2,3 | 2,0 | 2,0 |

Figure: Example from the original paper

## Complexity of the algorithm

- running-time: O(nm)
  - algorithm is exact, but very time consuming.
  - FASTA is an heuristic approximation and mostly used today.
- need of space: O(nm)

# Smith-Waterman algorithm
## Disadvantages



- time and space cost are very high
- finds the alignment with maximal score, but not with maximal percent of matches
  - algorithm makes 'mosaics' of well-conserved fragments with connections by poorly-conserved fragments
  - solution: length-normalized local alignment

$\rightarrow$ *obtains the region with maximum degree of similarity*

# Smith-Waterman algorithm
## Applications



- JAligner
- SSEARCH (in FASTA package)
- Live-Demo of the Smith-Waterman algorithm: http://baba.sourceforge.net/

# Bibliography



[1] Alison Cawsey, Dynamic Programming, http://www.macs.hw.ac.uk/ãlison/ds98/node122.html, 1998

[2] Temple F. Smith and Michael S. Waterman, Identification of Common Molecular Subsequences, J. Mol. Biol., 147(1):195-197, March 1981

[3] Script: Sequence Analysis I+II, Lecture notes Faculty of Technology, Bielefeld University, Winter 2008/09 and Summer 2009

[4] Norman Casagrande, Basic-Algorithms of Bioinformatics Applet, http://baba.sourceforge.net/, 2003

[5] University of Southern California, University Professor, http://www.cmb.usc.edu/people/msw/Waterman.html, 2005

# Thank you!



## The End

*Thank you for your attention!*