Politecnico di Bari

# Politecnico di Bari

Dipartimento di Ingegneria Elettrica e dell'Informazione

Corso di Laurea Magistrale in Ingegneria dei Sistemi Medicali

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE

# Bioinformatica Avanzata

## Genetic Algorithms for Multiple Sequence Alignment (with MATLAB examples)

Dr. Nicola **Altini**, Ph.D. Student

Prof. Eng. Vitoantonio **Bevilacqua**, Ph.D.

INDUSTRIAL INFORMATICS LAB

LABORATORIO DI INFORMATICA INDUSTRIALE

Anno Accademico 2019/2020

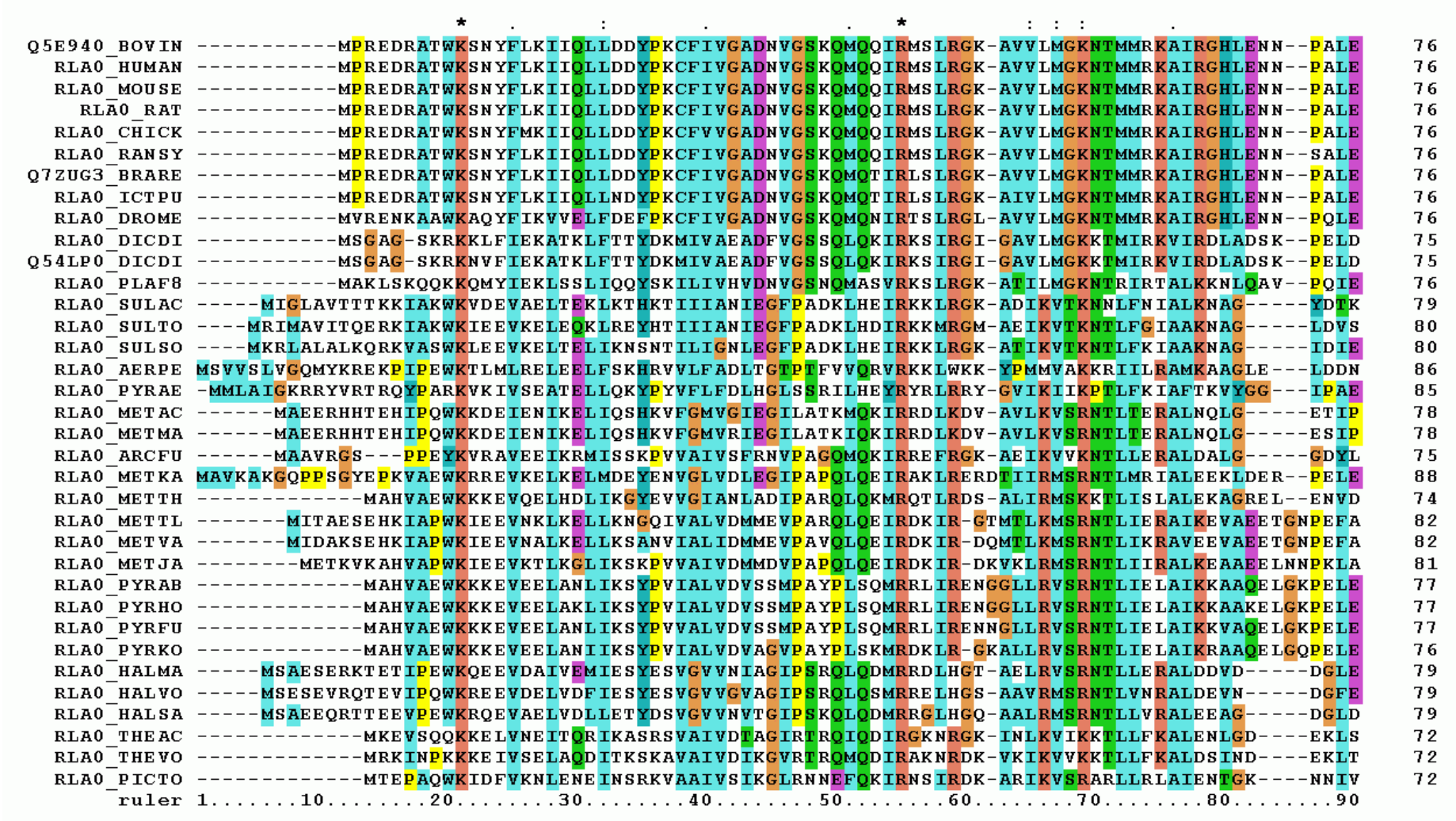
apulian bioengineering company

# Multiple Sequence Alignment

- A **multiple sequence alignment** (**MSA**) is a sequence alignment of three or more biological sequences, generally protein, DNA, or RNA. In many cases, the input set of query sequences are assumed to have an evolutionary relationship by which they share a linkage and are descended from a common ancestor.
- From the resulting MSA, sequence homology can be inferred and phylogenetic analysis can be conducted to assess the sequences' shared evolutionary origins. Multiple sequence alignment is often used to assess sequence conservation of protein domains, tertiary and secondary structures, and even individual amino acids or nucleotides.

# Multiple Sequence Alignment

Visual depictions of the alignment as in the image at right illustrate mutation events such as point mutations (single amino acid or nucleotide changes) that appear as differing characters in a single alignment column, and insertion or deletion mutations (indels or gaps) that appear as hyphens in one or more of the sequences in the alignment.



# Multiple Sequence Alignment

- Multiple sequence alignment also refers to the process of aligning such a sequence set. Because three or more sequences of biologically relevant length can be difficult and are almost always time-consuming to align by hand, computational algorithms are used to produce and analyze the alignments.
- MSAs require more sophisticated methodologies than pairwise alignment because they are more computationally complex. Most multiple sequence alignment programs use heuristic methods rather than global optimization because identifying the optimal alignment between more than a few sequences of moderate length is prohibitively computationally expensive.

# Mathematical Definition

- Given $m$ sequences $S_i, i = 1, \dots, m$ similar to the form below:

$$S := \begin{cases} S_1 = (S_{11}, S_{12}, \dots, S_{1n_1}) \\ S_2 = (S_{21}, S_{22}, \dots, S_{2n_2}) \\ \dots \\ S_m = (S_{m1}, S_{m2}, \dots, S_{mn_m}) \end{cases}$$

- We want to obtain $m$ modified sequences $S'_i$, with the following requirements:
  - All these sequence must conform to the length $L \geq \max\{\, length(S_i) \mid i = 1, \dots, m\}$
  - No column can be made of only gaps
- The mathematical form of an MSA of the sequence set $S$ is:

$$S' := \begin{cases} S'_1 = (S'_{11}, S'_{12}, \dots, S'_{1L}) \\ S'_2 = (S'_{21}, S'_{22}, \dots, S'_{2L}) \\ \dots \\ S'_m = (S'_{m1}, S'_{m2}, \dots, S'_{mL}) \end{cases}$$

Given an $S'_i$, it is possible to obtain $S_i$ by removing all the gaps.

# Mathematical Definition

- Definition. **Multiple alignment**.
- A **multiple (global) alignment** of $k$ sequences, is an assignment of gap symbols "–" into those sequences, or at their ends. The $k$ resulting strings are placed one above the other so that every character or gap symbol in either string is opposite a unique character or a unique gap symbol in the other string. It can be represented as a $c \times k$ matrix, for some value of $c$, the $i$th row containing the $i$th sequence and (interspersed) gap symbols.
- From a biological perspective, a multiple alignment represents a hypothesis about homology of individual positions within the aligned sequences.

# Example

```
ATTCCGA
TCCGTA
AAACCG
TTCCA
GACCTT
```


```
ATTCCG-A
--TCCGTA
AAACCG--
-TTCC-A-
-GACCTT-
```

**Objective Function: 96**

# Needleman–Wunsch algorithm

- The Needleman–Wunsch algorithm is an algorithm used in bioinformatics to align protein or nucleotide sequences. It was one of the first applications of dynamic programming to compare biological sequences. The algorithm was developed by Saul B. Needleman and Christian D. Wunsch and published in 1970.
- The algorithm essentially divides a large problem (e.g. the full sequence) into a series of smaller problems, and it uses the solutions to the smaller problems to find an optimal solution to the larger problem.
- It is also sometimes referred to as the optimal matching algorithm and the global alignment technique. The Needleman–Wunsch algorithm is still widely used for optimal global alignment, particularly when the quality of the global alignment is of the utmost importance. The algorithm assigns a score to every possible alignment, and the purpose of the algorithm is to find all possible alignments having the highest score.

# Needleman–Wunsch algorithm

match = 1 mismatch = -1 gap = -1

| | | G | C | A | T | G | C | U |
|---|---|---|---|---|---|---|---|---|
| | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 |
| G | -1 | 1 | 0 | -1 | -2 | -3 | -4 | -5 |
| A | -2 | 0 | 0 | 1 | 0 | -1 | -2 | -3 |
| T | -3 | -1 | -1 | 0 | 2 | 1 | 0 | -1 |
| T | -4 | -2 | -2 | -1 | 1 | 1 | 0 | -1 |
| A | -5 | -3 | -3 | -1 | 0 | 0 | 0 | -1 |
| C | -6 | -4 | -2 | -2 | -1 | -1 | 1 | 0 |
| A | -7 | -5 | -3 | -1 | -2 | -2 | 0 | 0 |

# BLOSUM

- The **BLOSUM** (**BLO**cks **SU**bstitution **M**atrix) matrix is a substitution matrix used for sequence alignment of proteins. BLOSUM matrices are used to score alignments between evolutionarily divergent protein sequences. They are based on local alignments. BLOSUM matrices were first introduced in a paper by Steven Henikoff and Jorja Henikoff.
- They scanned the BLOCKS database for very conserved regions of protein families (that do not have gaps in the sequence alignment) and then counted the relative frequencies of amino acids and their substitution probabilities.
- Then, they calculated a log-odds score for each of the 210 possible substitution pairs of the 20 standard amino acids. All BLOSUM matrices are based on observed alignments; they are not extrapolated from comparisons of closely related proteins like the PAM Matrices.

# BLOSUM62

```
# Entries for the BLOSUM62 matrix at a scale of ln(2)/2.0.
```

| | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | B | J | Z | X | * |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **A** | 4 | -1 | -2 | -2 | 0 | -1 | -1 | 0 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 0 | -3 | -2 | 0 | -2 | -1 | -1 | -1 | -4 |
| **R** | -1 | 5 | 0 | -2 | -3 | 1 | 0 | -2 | 0 | -3 | -2 | 2 | -1 | -3 | -2 | -1 | -1 | -3 | -2 | -3 | -1 | -2 | 0 | -1 | -4 |
| **N** | -2 | 0 | 6 | 1 | -3 | 0 | 0 | 0 | 1 | -3 | -3 | 0 | -2 | -3 | -2 | 1 | 0 | -4 | -2 | -3 | 4 | -3 | 0 | -1 | -4 |
| **D** | -2 | -2 | 1 | 6 | -3 | 0 | 2 | -1 | -1 | -3 | -4 | -1 | -3 | -3 | -1 | 0 | -1 | -4 | -3 | -3 | 4 | -3 | 1 | -1 | -4 |
| **C** | 0 | -3 | -3 | -3 | 9 | -3 | -4 | -3 | -3 | -1 | -1 | -3 | -1 | -2 | -3 | -1 | -1 | -2 | -2 | -1 | -3 | -1 | -3 | -1 | -4 |
| **Q** | -1 | 1 | 0 | 0 | -3 | 5 | 2 | -2 | 0 | -3 | -2 | 1 | 0 | -3 | -1 | 0 | -1 | -2 | -1 | -2 | 0 | -2 | 4 | -1 | -4 |
| **E** | -1 | 0 | 0 | 2 | -4 | 2 | 5 | -2 | 0 | -3 | -3 | 1 | -2 | -3 | -1 | 0 | -1 | -3 | -2 | -2 | 1 | -3 | 4 | -1 | -4 |
| **G** | 0 | -2 | 0 | -1 | -3 | -2 | -2 | 6 | -2 | -4 | -4 | -2 | -3 | -3 | -2 | 0 | -2 | -2 | -3 | -3 | -1 | -4 | -2 | -1 | -4 |
| **H** | -2 | 0 | 1 | -1 | -3 | 0 | 0 | -2 | 8 | -3 | -3 | -1 | -2 | -1 | -2 | -1 | -2 | -2 | 2 | -3 | 0 | -3 | 0 | -1 | -4 |
| **I** | -1 | -3 | -3 | -3 | -1 | -3 | -3 | -4 | -3 | 4 | 2 | -3 | 1 | 0 | -3 | -2 | -1 | -3 | -1 | 3 | -3 | 3 | -3 | -1 | -4 |
| **L** | -1 | -2 | -3 | -4 | -1 | -2 | -3 | -4 | -3 | 2 | 4 | -2 | 2 | 0 | -3 | -2 | -1 | -2 | -1 | 1 | -4 | 3 | -3 | -1 | -4 |
| **K** | -1 | 2 | 0 | -1 | -3 | 1 | 1 | -2 | -1 | -3 | -2 | 5 | -1 | -3 | -1 | 0 | -1 | -3 | -2 | -2 | 0 | -3 | 1 | -1 | -4 |
| **M** | -1 | -1 | -2 | -3 | -1 | 0 | -2 | -3 | -2 | 1 | 2 | -1 | 5 | 0 | -2 | -1 | -1 | -1 | -1 | 1 | -3 | 2 | -1 | -1 | -4 |
| **F** | -2 | -3 | -3 | -3 | -2 | -3 | -3 | -3 | -1 | 0 | 0 | -3 | 0 | 6 | -4 | -2 | -2 | 1 | 3 | -1 | -3 | 0 | -3 | -1 | -4 |
| **P** | -1 | -2 | -2 | -1 | -3 | -1 | -1 | -2 | -2 | -3 | -3 | -1 | -2 | -4 | 7 | -1 | -1 | -4 | -3 | -2 | -2 | -3 | -1 | -1 | -4 |
| **S** | 1 | -1 | 1 | 0 | -1 | 0 | 0 | 0 | -1 | -2 | -2 | 0 | -1 | -2 | -1 | 4 | 1 | -3 | -2 | -2 | 0 | -2 | 0 | -1 | -4 |
| **T** | 0 | -1 | 0 | -1 | -1 | -1 | -1 | -2 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 5 | -2 | -2 | 0 | -1 | -1 | -1 | -1 | -4 |
| **W** | -3 | -3 | -4 | -4 | -2 | -2 | -3 | -2 | -2 | -3 | -2 | -3 | -1 | 1 | -4 | -3 | -2 | 11 | 2 | -3 | -4 | -2 | -2 | -1 | -4 |
| **Y** | -2 | -2 | -2 | -3 | -2 | -1 | -2 | -3 | 2 | -1 | -1 | -2 | -1 | 3 | -3 | -2 | -2 | 2 | 7 | -1 | -3 | -1 | -2 | -1 | -4 |
| **V** | 0 | -3 | -3 | -3 | -1 | -2 | -2 | -3 | -3 | 3 | 1 | -2 | 1 | -1 | -2 | -2 | 0 | -3 | -1 | 4 | -3 | 2 | -2 | -1 | -4 |
| **B** | -2 | -1 | 4 | 4 | -3 | 0 | 1 | -1 | 0 | -3 | -4 | 0 | -3 | -3 | -2 | 0 | -1 | -4 | -3 | -3 | 4 | -3 | 0 | -1 | -4 |
| **J** | -1 | -2 | -3 | -3 | -1 | -2 | -3 | -4 | -3 | 3 | 3 | -3 | 2 | 0 | -3 | -2 | -1 | -2 | -1 | 2 | -3 | 3 | -3 | -1 | -4 |
| **Z** | -1 | 0 | 0 | 1 | -3 | 4 | 4 | -2 | 0 | -3 | -3 | 1 | -1 | -3 | -1 | 0 | -1 | -2 | -2 | -2 | 0 | -3 | 4 | -1 | -4 |
| **X** | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -4 |
| ***** | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | 1 |

# Computational Cost

- The extension of pairwise alignment algorithms such as Needleman–Wunsch or Smith–Waterman to more than two sequences are straightforward, but their cost increases exponentially with $k$ (where $k$ is the number of sequences).
- This fact means that there is an exponential cost in the number of sequences being aligned, limiting the use of these algorithms to very few sequences.
- Computational cost for two sequences of length $n$: $O(n^2)$
- Computational cost for $k$ sequences of length $n$: $O(n^k)$
  - The cost is exponential in $k$
- To find the global optimum for n sequences this way has been shown to be an NP-complete problem.

# Genetic Algorithms for MSA

- Genetic algorithms, has been used for MSA production in an attempt to broadly simulate the hypothesized evolutionary process that gave rise to the divergence in the query set.
- The method works by breaking a series of possible MSAs into fragments and repeatedly rearranging those fragments with the introduction of gaps at varying positions.
- A general objective function is optimized during the simulation, most generally the "sum of pairs" maximization function introduced in dynamic programming-based MSA methods.
- A technique for protein sequences has been implemented in the software program **SAGA** (**S**equence **A**lignment by **G**enetic **A**lgorithm) and its equivalent in RNA is called **RAGA**.

# Genetic Algorithms for MSA

- Objective Function
  - Sum of pairs (SP) score
- Crossover Operation
  - Take some sequences from parent 1
  - Take some sequences from parent 2
  - Merge and get child
- Mutation Operation
  - Adding or removing random gaps from child
- Util operations
  - Gaps padding
  - Gaps columns deletion

# Algorithm Overview - SAGA

- Population made of alignments. Population size is constant.
- Generation zero $G_0$ is randomly created.
- To go from one generation to the next, children are derived from parents using natural selection, based on fitness of parents measured by the OF.
- Child can be either the result of a crossover (mixing the contents of the two parents) or a mutation (modifying a single parent).
- Each operator has a probability of being chosen which is dynamically optimized during the execution.
- Repeat these steps iteratively.

# Algorithm Overview

- Initialization:
  - `PrepareInput()`
  - `InitializePop()`
- Optimization:
  - `While generation < numGenerations`
    - `EvalFunction()`
    - `SelectParents()`
    - `ApplyCrossover()`
    - `ApplyMutation()`
    - `GapsPadding()`
    - `GapsColumnsDeletion()`
    - `CheckNoChanges()`
    - `UpdatePopulation()`

# Initializing population

- Compute pairwise alignments between all the original pairs.
- Random sample these alignments to create the desired number of chromosomes.

```
{["ATTCCGCAGTCGGACTACGTACGA--------"]}
{["TCCGTACAGTCGCTAGCATCGAT---------"]}
{["AAACCGCGCGCGTCGTCAGA------------"]}
{["TTCCACAGTCGATCGCA---------------"]}
{["GACCTTCGATCGACGACGATCGGCATGAGTCA"]}
{["CAGTCAGCTACGACTGAT--------------"]}
{["CAGCTAGCTTCAGTCAG---------------"]}
{["CGACGACTACGCGTA-----------------"]}
```

Sequences from original file

```
"TCCGTACAGTCGCTAGCATCGAT"
"ATTCCGCAGTCGGACTACGTACGA"
```

1. Select two sequences

```
"--TCCGTACAGTC-G-CTAGCAT-CGAT---------"
"  ||||  |||||  |  |||  |:|  |||       "
"ATTCCG--CAGTCGGACTA-CGTACGA----------"
```

2. Compute alignment between the sequences (for instance, with NW algorithm).

```
"--TCCGTACAGTC-G-CTAGCAT-CGAT---------"
```

3. Take the first row of the alignment as a chromosome for the initial population

# Initializing population

**Input Sequences**

```
{["ATTCCGA"]}
{["TCCGTA-"]}
{["AAACCG-"]}
{["TTCCA--"]}
{["GACCTT-"]}
```

**Chromosome 1 on 8:**

```
["ATTCCGA--"]
["--TCCGTA-"]
["AAACCG---"]
["TTCCA----"]
["GACCTT---"]
```

**Chromosome 2 on 8:**

```
["ATTCCG-A-"]
["-TCCGTA--"]
["AAACCG---"]
["TTCCA----"]
["GACCTT---"]
```

**Chromosome 3 on 8:**

```
["ATTCCG-A"]
["-TCCGTA-"]
["AAACCG--"]
["-TTCCA--"]
["GACCTT--"]
```

**Chromosome 4 on 8:**

```
["ATTCCG-A-"]
["--TCCGTA-"]
["AAACCG---"]
["-TTCCA---"]
["GACCTT---"]
```

**Chromosome 5 on 8:**

```
["ATTCCGA-"]
["-TCCGTA-"]
["AAACCG--"]
["TTCCA---"]
["GACCTT--"]
```

**Chromosome 6 on 8:**

```
["ATTCCGA-"]
["-TCCGTA-"]
["AAACCG--"]
["TTCCA---"]
["GACCTT--"]
```

**Chromosome 7 on 8:**

```
["ATTCCGA-"]
["-TCCGTA-"]
["AAACCG--"]
["-TTCCA--"]
["GACCTT--"]
```

**Chromosome 8 on 8:**

```
["ATTCCGA-"]
["-TCCGTA-"]
["AAACCG--"]
["-TTCCA--"]
["GACCTT--"]
```

# Objective Function - SAGA

- Global cost made up of:
  - Substitution cost (cost to each pair of aligned residues)
  - Gaps cost
- Each pair of sequences is given a weight related to their similarity to the other pairs.
- Variations:
  i. Using different sets of sequence weights
  ii. Different sets of substitution costs (PAM or BLOSUM tables)
  iii. Different schemas for scoring of gaps
- The cost for a multiple alignment $S$ is then:

$$AlignmentCost(S) = \sum_{i=2}^{N} \sum_{j=1}^{i-1} W_{ij} Cost(S_i, S_j)$$

# Objective Function

- The objective function to maximize is the sum of all pairwise scores calculated between the sequences in a chromosome.
- Pairwise scores are calculated using Needleman–Wunsch algorithm.

```
["ATTCCGA-"]
["--TCCGTA"]
["AAACCG--"]
["TTCCA---"]
["GACCTT--"]
```

**Objective Function**
**34**

```
["ATTCCGA-"]
["-TCCGTA-"]
["AAACCG--"]
["TTCCA---"]
["GACCTT--"]
```

**Objective Function**
**34.25**

# Objective Function

```
function [sum_score] = eval_function(chromosome)
        sum_score = 0;
        for i = 2:size(chromosome,1)
           % Compute pairwise scores
           for j = 1:(i-1)
               [curr_score, ~] = nwalign(chromosome{i,1}, chromosome{j,1}, ...
                                         'ScoringMatrix', 'BLOSUM62', ...
                                         'GapOpen', 1, ...
                                         'ExtendGap', 0.5);
               sum_score = sum_score + curr_score;
           end
      end
end
```

# Crossover Operation

**Parent 1:**

```
["ATTCCGA--"]
["--TCCGTA-"]
["AAACCG---"]
["-TTCCA---"]
["GACCTT---"]
```

**Parent 2:**

```
["ATTCCGA-"]
["-TCCGTA-"]
["AAACCG--"]
["TTCCA---"]
["GACCTT--"]
```

**Child:**

```
["ATTCCGA--"]
["--TCCGTA-"]
["AAACCG--" ]
["TTCCA---" ]
["GACCTT--" ]
```

# Gaps padding

**Child:**

```
["ATTCCGA--"]
["--TCCGTA-"]
["AAACCG--" ]
["TTCCA---" ]
["GACCTT--" ]
```

**Child:**

```
["ATTCCGA--"]
["--TCCGTA-"]
["AAACCG---"]
["TTCCA----"]
["GACCTT---"]
```

# Gaps columns deletion

**Child:**

```
["ATTCCGA--"]
["--TCCGTA-"]
["AAACCG---"]
["TTCCA----"]
["GACCTT---"]
```

**Child:**

```
["ATTCCGA-"]
["--TCCGTA"]
["AAACCG--"]
["TTCCA---"]
["GACCTT--"]
```

# Mutation Operation

**Child before mutation:**

```
["ATTCCGA--"]
["--TCCGTA-"]
["AAACCG---"]
["TTCCA----"]
["GACC-TT--"]
```

**Child after mutation:**

```
["ATTCCGA--" ]
["--TCCGTA-" ]
["AAACCG----"]
["TTCCA----" ]
["GACC-TT--" ]
```

**Child after gaps padding:**

```
["ATTCCGA---"]
["--TCCGTA--"]
["AAACCG----"]
["TTCCA-----"]
["GACC-TT---"]
```

**Child after gaps deletion:**

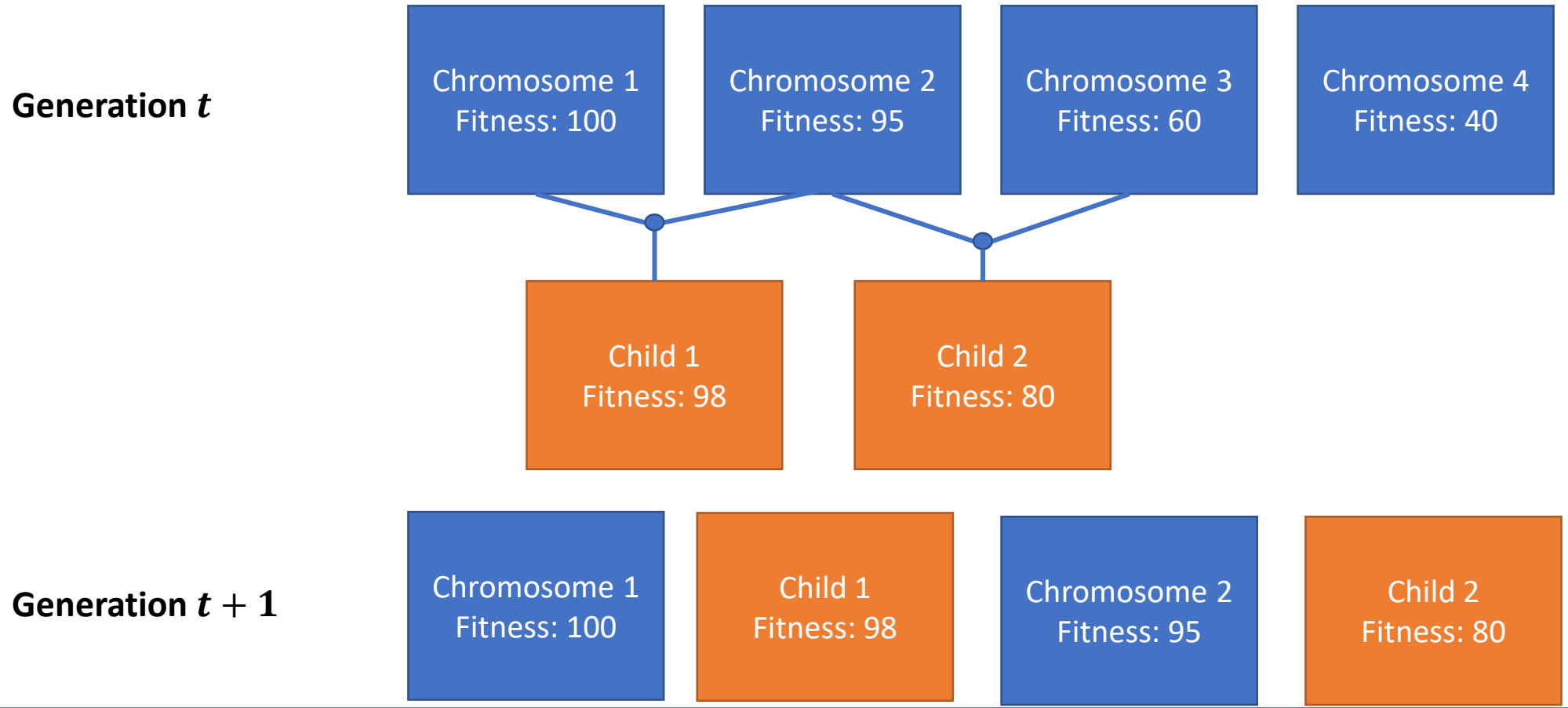
```
["ATTCCGA-"]
["--TCCGTA"]
["AAACCG--"]
["TTCCA---"]
["GACC-TT-"]
```

# Optimization parameters

- `chromosomes`
  - The number of chromosomes which will compose the population
- `generations`
  - The number of (maximum) generations for which the algorithm will try to optimize the objective function.
- `min_num_gen`
  - The minimum number of generations before checking if there are no more progresses.
- `crossover_prob`
  - The probability for a crossover to happen.
- `mutation_rate`
  - The probability for a mutation to happen.

# Update population

- Example: population of 4 chromosomes.
  - *Probability of crossover*: 0.5
  - *Probability of mutation*: 0.05



# Stopping heuristic

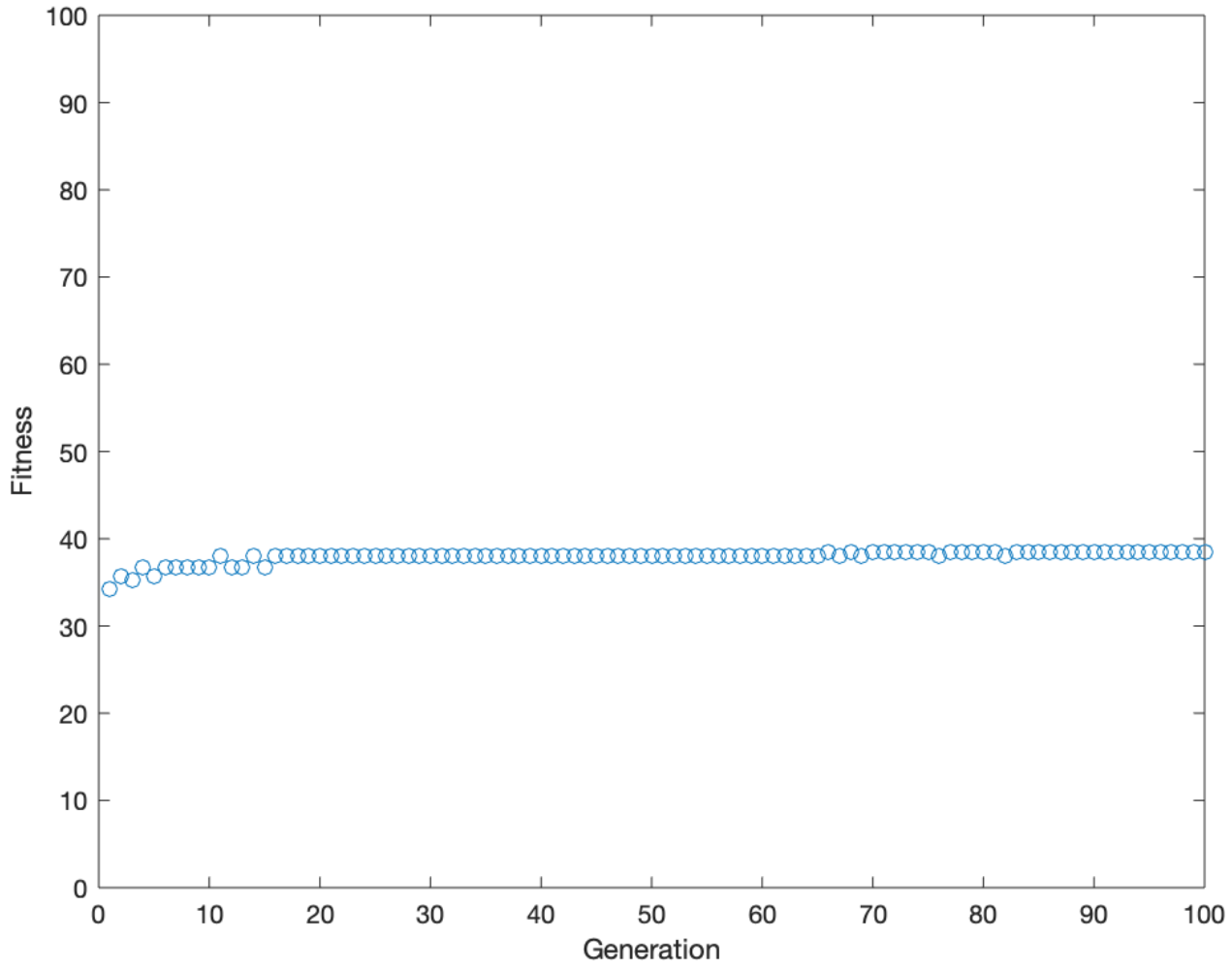
- Calculate the variance in a sample of last best fitness values.
- If variance is below a threshold, then assess that there is no more progress in the optimization process and stop the algorithm.

```
last_best_values
    35.2500
    35.2500
    35.2500
    35.2500
    35.2500
    35.2500
    35.2500
    35.2500
    35.2500
    35.2500
```

`var(last_best_values) = 0`

**Optimization stop**

# Optimization Process



# Optimization Result

- Results of the optimization:

**Original sequences**

```
["ATTCCGA"]
["TCCGTA" ]
["AAACCG" ]
["TTCCA"  ]
["GACCTT" ]
```

**Best chromosome**

```
["ATTCCGA-"]
["--TCCGTA"]
["AAACCG--"]
["TTCCA---"]
["GA-CC-TT"]
```

**Fitness Function**

38.50

# References


- Github
  - Multiple Sequence Alignment - Genetic Algorithm
    URL: https://github.com/filipefalcaos/msa-ga
- MATLAB Documentation
  - *Needleman-Wunsch algorithm.*
    URL: https://it.mathworks.com/help/bioinfo/ref/nwalign.html
- Wikipedia
  - Multiple Sequence Alignment.
    URL: https://en.wikipedia.org/wiki/Multiple_sequence_alignment
  - *Needleman-Wunsch algorithm.*
    URL: https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm
  - BLOSUM.
    URL: https://en.wikipedia.org/wiki/BLOSUM
- Notredame, C., & Higgins, D. G. (1996). SAGA: sequence alignment by genetic algorithm. Nucleic acids research, 24(8), 1515-1524.
- Notredame, C., O'Brien, E. A., & Higgins, D. G. (1997). RAGA: RNA sequence alignment by genetic algorithm. *Nucleic acids research*, *25*(22), 4570-4580.
- Cristianini, N., & Hahn, M. W. (2006). *Introduction to computational genomics: a case studies approach*. Cambridge University Press.