Application to the comparison of sequences

GENSTORM

Tutorial

A specialised architecture for sequences processing

Emeka MOSANYA

Logic Systems Laboratory

Swiss Federal Institute of Technology

Introduction

This tutorial is intended to provide you with :

- Basic knowledge of exhaustive sequence comparisons
- Their implementation with a systolic architecture
- The example of the GENSTORM machine currently in development

If you require further information you should :

- Refer to the bibliography
- Consult the GENSTORM Web Server to follow the advance of the project and find additional links.

Table of contents

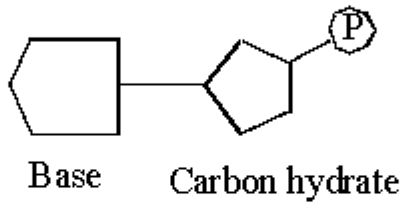
Table of contents

Biology reminder

- The majority of living beings is composed of cells
- There is a high degree of similarity between cells from different species
- A cell is composed of :
  - 75 to 80 % of water
  - 10 to 20 % of nucleic acids and proteins
  - 2 to 3 % of lipids
  - 1 % of carbohydrates
  - 1 % of inorganic elements
- A human organism is composed of about 10^{15} cells
- The structure and the function of a cell, as well as those of a complete organism, are coded in DNA molecules.

Biology reminder

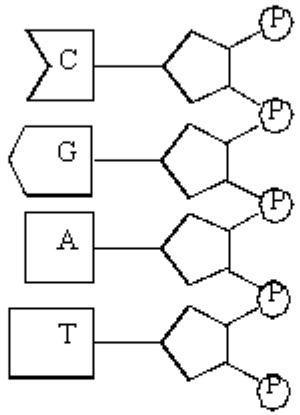
- The DNA is a molecule composed of mononucleotides, themselves composed of a phosphorus atom, a carbon hydrate, and a base :



- The first two elements are structural.
- The third is used to code information. In the DNA molecule, a base can be one of the following molecules : Adenine, Guanine, Cytosine and Thymine

Biology reminder

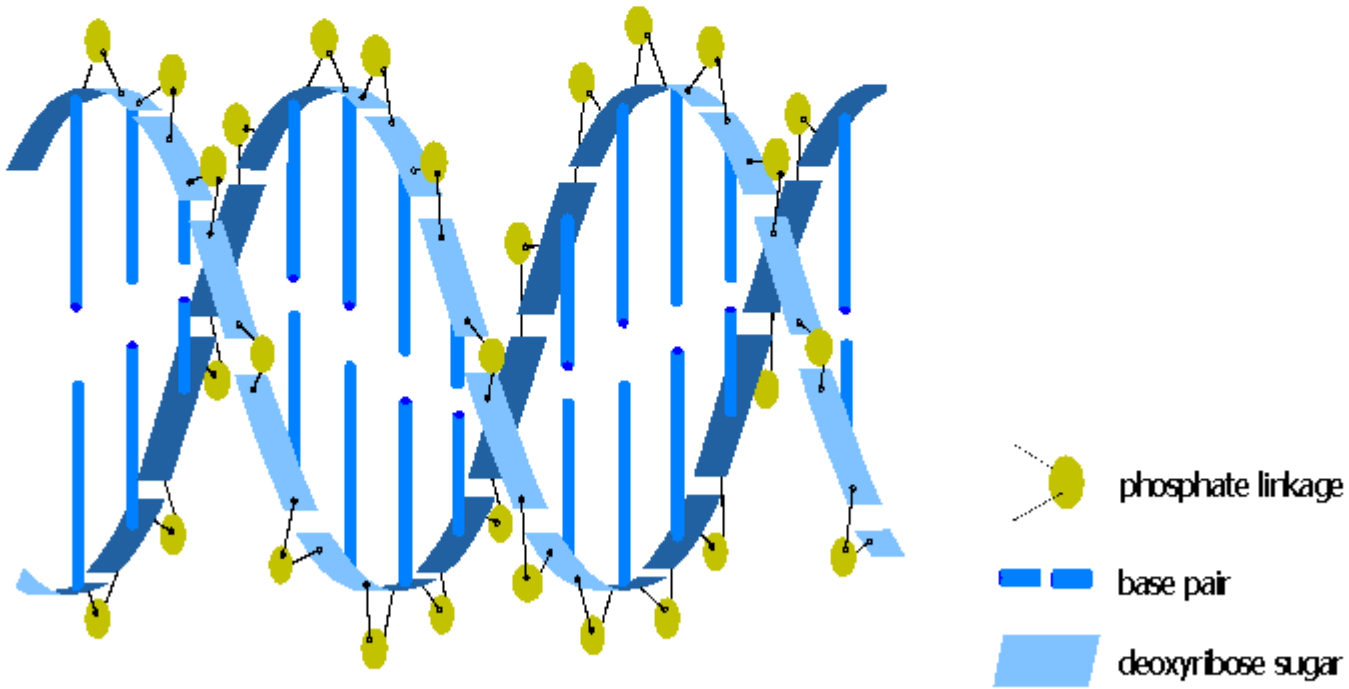
- The mononucleotides chain together to form a polynucleotide :



- The letters A, G, C, and T represent the four elements mentioned above : Adenine, Guanine, Cytosine and Thymine

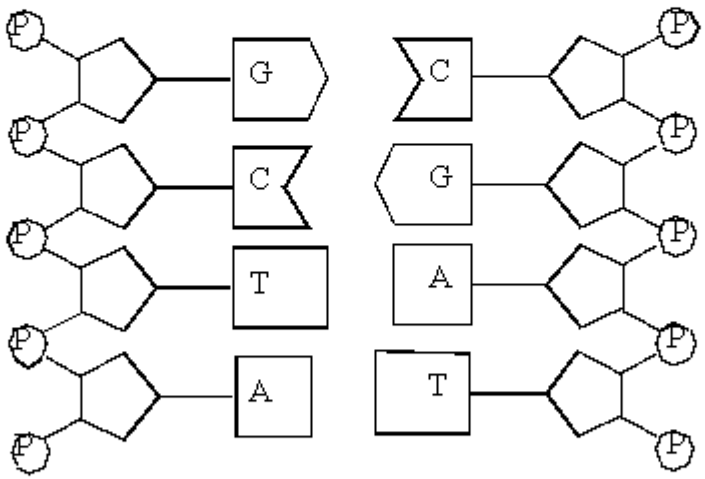
Biology reminder

- The structure of the DNA was discovered by Watson and Crick in 1953 [WAT53] : two polynucleotide chains linked together to form a double helix.



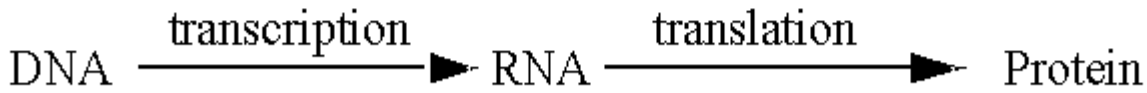
Biology reminder

- Adenine can only match thymine and guanine can only match cytosine; therefore, the two chains are complementary and only one is needed to carry all the information.



Biology reminder

- It is estimated that only 10% of the DNA code contains useful information.
- A gene is a coding sequence of DNA used by the cells. The set of genes forms the genome.
- The proteins are the basic building blocks of living beings. They are composed of simpler elements called amino-acids. The assembly of these amino-acids determines the structure and therefore the function of a protein.
- The protein synthesis is the result of a two-step process :
  - A DNA sequence is copied by a RNA molecule.
  - The RNA molecule is decoded in the ribosome to generate a protein.



Biology reminder

- Thus, all the information needed for protein synthesis is contained in the DNA, coded by the succession of the four types of base.
- Since there are 20 different types of amino-acid, a minimum of three bases are needed to code one amino-acid. This three-base sequence is called codon.
- By definition, a codon can code up to 64 different amino-acids : there are several ways to code a single amino-acid and some codons are used as control sequences.

Biology reminder conclusion

- Living beings are composed of cells.
- The specific cell functions are carried out with the help of numerous proteins.
- The structure of proteins is coded by the genome.
- Studying the genome, i.e. its biochemical mechanism and the information it contains, is a fundamental task in the quest to improve our knowledge of the mechanisms of life.

Databases

- As a consequence of a world-wide research effort, a considerable amount of data is being generated.
- These data are stored in several databases :
  - Genome : humans, bacteria, yeast, flies…
  - Proteins : SwissProt,…
  - Patterns : Prosite,…
  - …
- These data need processing tools
  - BLAST : Heuristic sequence comparison
  - ScanProsite,…
  - …
- The network is used intensively : Internet, WWW

Databases

- The size of the databases rises exponentially.

Basic sequence comparison

- Ignoring the biological properties of the sequences we process, we will consider the DNA, the RNA, and the proteins as chains of characters.
- Several methods have been used to compare sequences :
  - visual methods,
  - heuristic methods,
  - exhaustive methods.
- In this tutorial, we will focus on exhaustive methods since they are more effective if enough computing power is available.

Alignment of sequences

- A method which only computes exact matches between two sequences is not very interesting since two sequences corresponding to the same function can be slightly different due, for example, to mutations.
- To take these differences into account, we will compute an alignment as below :

```
 10        20          30        40        50
132955 ATGATGAACAGACGTCATTTTATTCAA---ATTAGTGCGACCAGTATTCTTGCATTAAGT
       :::         :::  :::    : :   ::: ::: ::: :  :::   ::    : :
_      ATG---------CGTTGTTTAGCACTAGATATTGGTGGGACAAAAATTGCAGCGGCGATT
                          10        20        30        40        50

        60                70          80        90
132955 GCAAAC-----CG---TTTTGCGATG---GCAAAAGGAAAGAGCG-------ATGTTGAT
       : :::      ::   ::  :: : :   :::::   : : : :        :::: : :
_      GTAAAAAATGGCGAAATTGAGCAACGTCAGCAAATTCATACACCACGTGAAAATGTCG-T
              60        70        80        90       100        110
```

Alignment of sequences

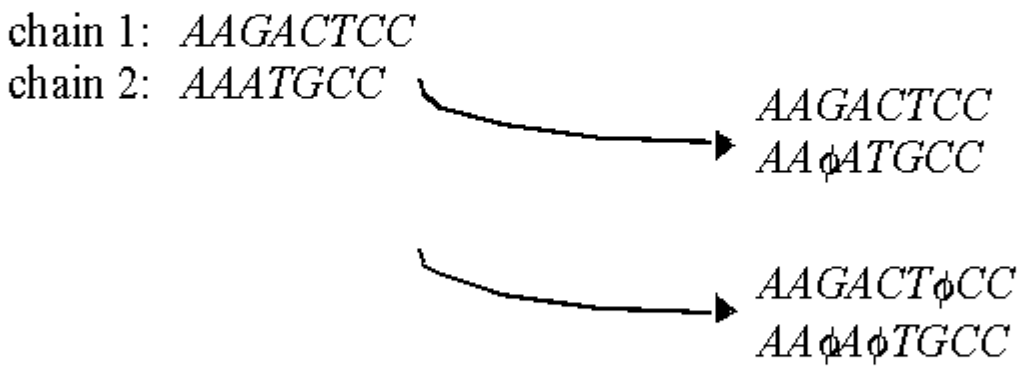
- As you can see in the figure below, an alignment tries to match two sequences, adding empty characters or enabling local mismatches.
- The resulting alignment depends on the definition used to compute it. For example, one can assume that adding a empty character or enabling a mismatch has a cost of one and that the best alignment is the one with the lowest score.

```
 10        20          30        40        50
132955 ATGATGAACAGACGTCATTTTATTCAA---ATTAGTGCGACCAGTATTCTTGCATTAAGT
       :::         :::  :::    : :   ::: ::: ::: :  :::   ::    : :
_      ATG---------CGTTGTTTAGCACTAGATATTGGTGGGACAAAAATTGCAGCGGCGATT
                          10        20        30        40        50

        60                70          80        90
132955 GCAAAC-----CG---TTTTGCGATG---GCAAAAGGAAAGAGCG-------ATGTTGAT
       : :::      ::   ::  :: : :   :::::   : : : :        :::: : :
_      GTAAAAAATGGCGAAATTGAGCAACGTCAGCAAATTCATACACCACGTGAAAATGTCG-T
              60        70        80        90       100        110
```

Alignment of sequences

- Let us consider a simpler example :



- ϕ represents the empty character

The distance between two sequences

- In this tutorial, we will focus on exhaustive methods.
- These methods are based on dynamic programming.
- Let a and b the two sequences, of length m and n, we want to compare.
- a^i and b^j denote the i^{th} base of the sequence and the j^{th} base of the sequence b, respectively.
- Let us define $d(a^i,b^j)$ as the distance between the two sub-sequences of length i+1 and j+1 beginning by a^0 and b^0, respectively.
- In this simple case, the distance will correspond to the minimal amount of deletion, insertion, substitution of bases we need to apply to transform one sequence to the other.

The distance between two sequences

- We define w_{supp}, w_{ins}, and w_{sub} as the costs associated to the deletion, the insertion, and the substitution of a base.
- We can define the following dynamic programming problem :

$$d(a^i,b^j) = \min\begin{cases} d(a^{i-1},b^j) + w_{supp} \\ d(a^{i-1},b^{j-1}) + w_{sub} \\ d(a^i,b^{j-1}) + w_{ins} \end{cases}$$

The distance between two sequences

- To resolve the recursion, we need some initial conditions :

$$d(\Phi,\Phi) = 0$$

$$d(\Phi,b^j) = d(\Phi,b^{j-1}) + w_{ins}$$

$$d(a^i,\Phi) = d(a^{i-1},\Phi) + w_{supp}$$

The distance between two sequences

- If we reuse the same example :

$$a = AAGACTCC$$

$$b = AAATGCC$$

$$w_{sub} = \begin{cases} 1 & \text{if the two bases are different} \\ 0 & \text{if the two bases are equal} \end{cases}$$

$$w_{supp} = 1$$

$$w_{ins} = 1$$

The distance between two sequences

- The calculation of the recursion can be represented by a matrix.
- To compute the value of one cell, we simply apply the formula :

$j \rightarrow$

$i \downarrow$

	A	A	A	T	G	C	C
A	0	1	2	3	4	5	6
A	1	0	1	2	3	4	5
G	2	1	1	2	2	3	4
A	3	2	1	2	3	3	4
C	4	3	2	?			
T							
C							
C							

$$d(a^5,b^4)=\min\begin{cases} d(a^4,b^4)+w(A,\phi) &= 2+1=3 \\ d(a^4,b^3)+w(C,T) &= 1+1=2 \\ d(a^5,b^3)+w(\phi,A) &= 2+1=3 \end{cases}$$

The distance between two sequences

- We continue the same process for each cell of the matrix until we obtain the final distance value :

$j \rightarrow$

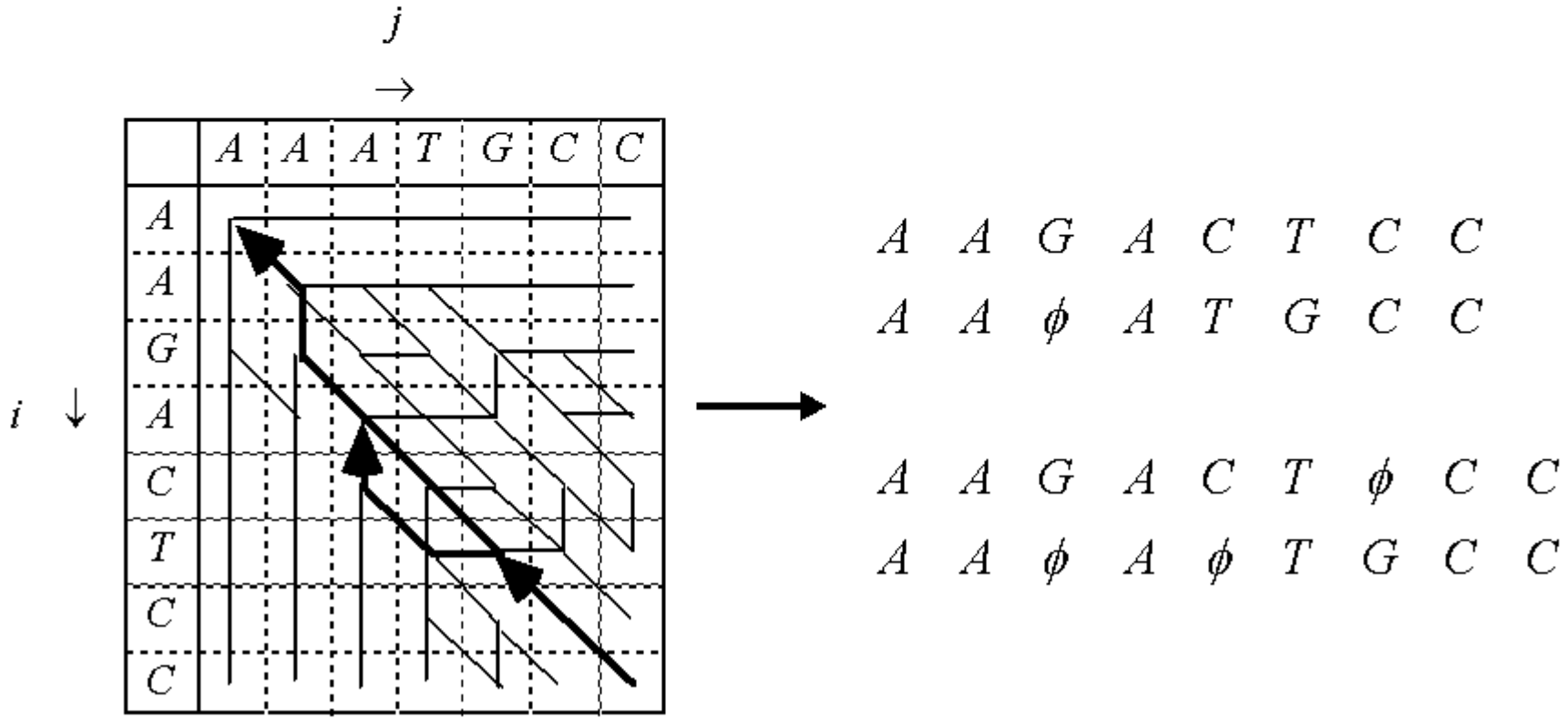
		A	A	A	T	G	C	C
	A	0	1	2	3	4	5	6
	A	1	0	1	2	3	4	5
	G	2	1	1	2	2	3	4
$i \downarrow$	A	3	2	1	2	3	3	4
	C	4	3	2	2	$\rightarrow$	$\rightarrow$	$\rightarrow$
	T	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	...		
	C							
	C							

$j \rightarrow$

		A	A	A	T	G	C	C
	A	0	1	2	3	4	5	6
	A	1	0	1	2	3	4	5
	G	2	1	1	2	2	3	4
$i \downarrow$	A	3	2	1	2	3	3	4
	C	4	3	2	2	3	3	3
	T	5	4	3	2	3	4	4
	C	6	5	4	3	3	3	4
	C	7	6	5	4	4	3	3

The alignment of two sequences

- The distance value does not give us the alignment itself.
- The compute the alignment, we need to track back from the final cell to the first, following each decision we have made during the evaluation of the minimum.



The alignment of two sequences

- In this example, the distance between the two sequences is 3.
- Following our definition of w_{supp}, w_{ins}, and w_{sub}, this means we need 3 operations to transform a into b.
- Since there is two paths resulting in a minimum distance, there is two possible alignments .

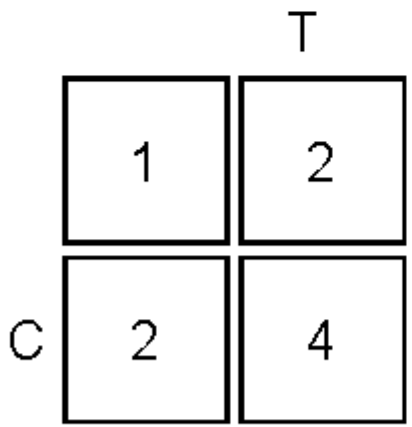
$$\begin{array}{cccccccc} A & A & G & A & C & T & C & C \\ A & A & \phi & A & T & G & C & C \end{array}$$

$$\begin{array}{ccccccccc} A & A & G & A & C & T & \phi & C & C \\ A & A & \phi & A & \phi & T & G & C & C \end{array}$$

Algorithm to compute the distance

- This is the corresponding algorithm :

Compute LeftVal = $V_l + W_{supp}$
Compute TopVal = $V_t + W_{ins}$
If $a^i = b^j$ **Then** DiagVal = V_{tl} Then DiagVal = $V_{tl} + W_{sub}$
Compute CellVal = **Min**(LeftVal,DiagVal,TopVal)



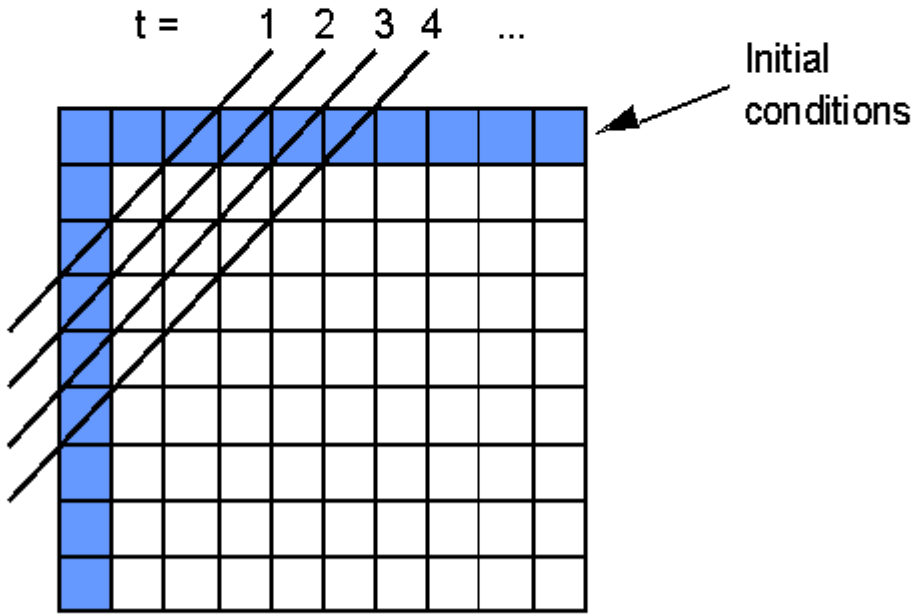
LeftVal = 2 + 1 = 3
TopVal = 2+ 1 = 3
T ≠ G then DiagVal = 1+ 1 = 2
CellVal = min(3,3,2) = 2

Algorithm complexity

- To compute the distance between two sequences we need to calculate each cell of the matrix :
  - Temporal complexity: O(n.m)
- To compute the value of a cell, we only need the left, top and diagonal values. So, we only need to memorise the equivalent of a row of the matrix :
  - Spatial complexity : O(n)

Temporal dependency

- We can not compute all the cell of the matrix simultaneously.
- At a point of time, only the cells along the corresponding diagonal can be computed.

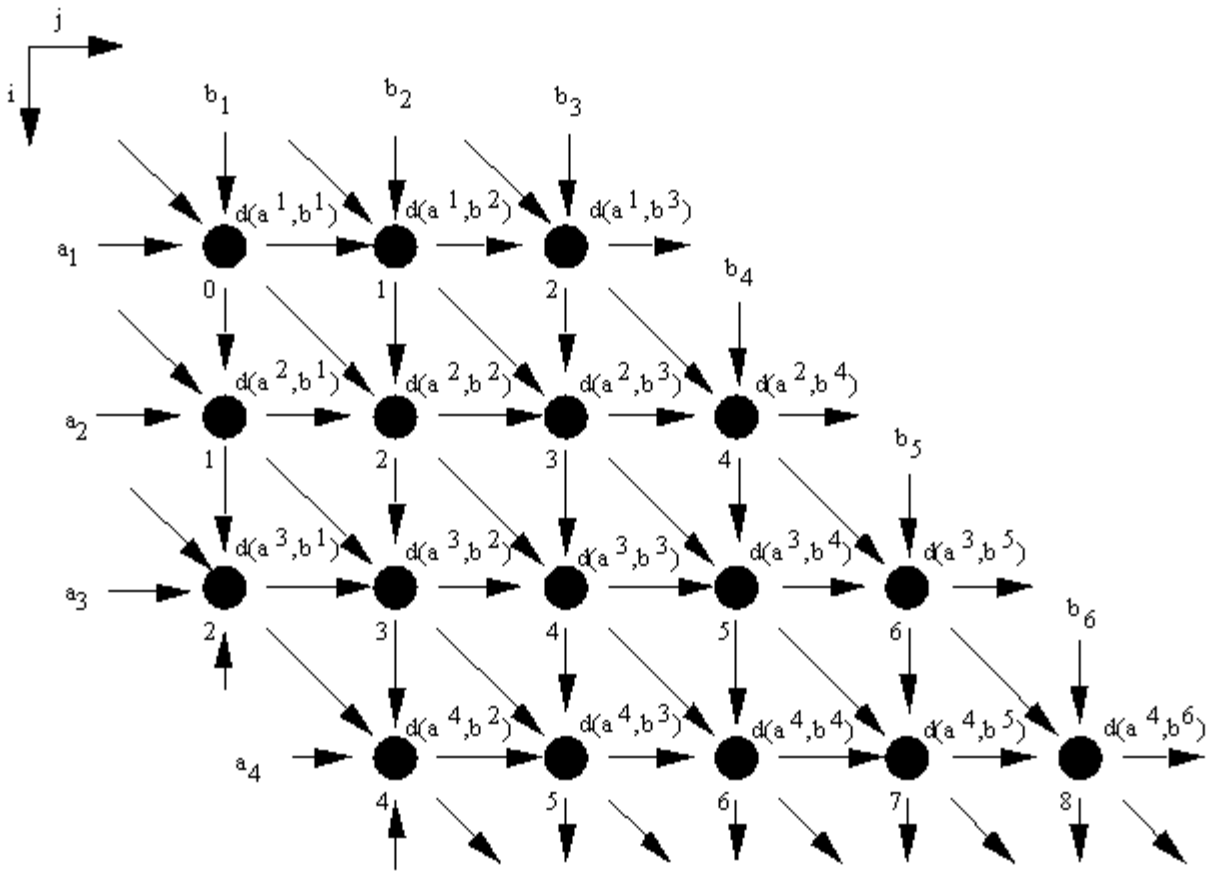


Specialised architecture

- In order to improve the performance of the comparison, we will take into account the specific properties of the algorithm used :
  - It is possible to divide the work in sub-tasks as we saw that all the cells on a diagonal can be evaluated simultaneously.
  - The evaluation of a cell is the same for all of them.
  - To evaluate the value of a cell, we only require the value of three neighbour cells.
  - The evaluation is constrained by the computation and not by the I/O throughput.
- As the result of these observations, a systolic architecture is the most adapted.

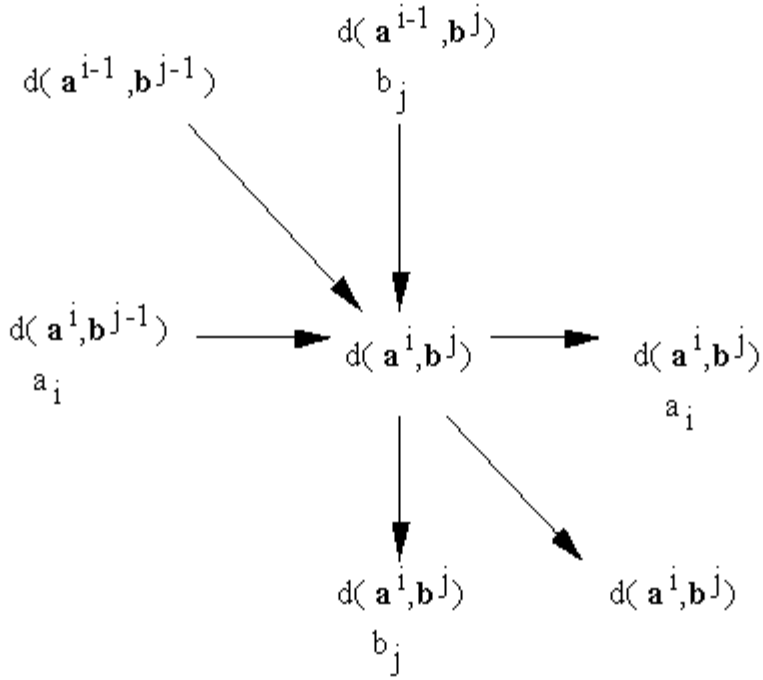
Application to the comparison of sequences

- Let us observe the data fluxes and dependencies in the preceding example:



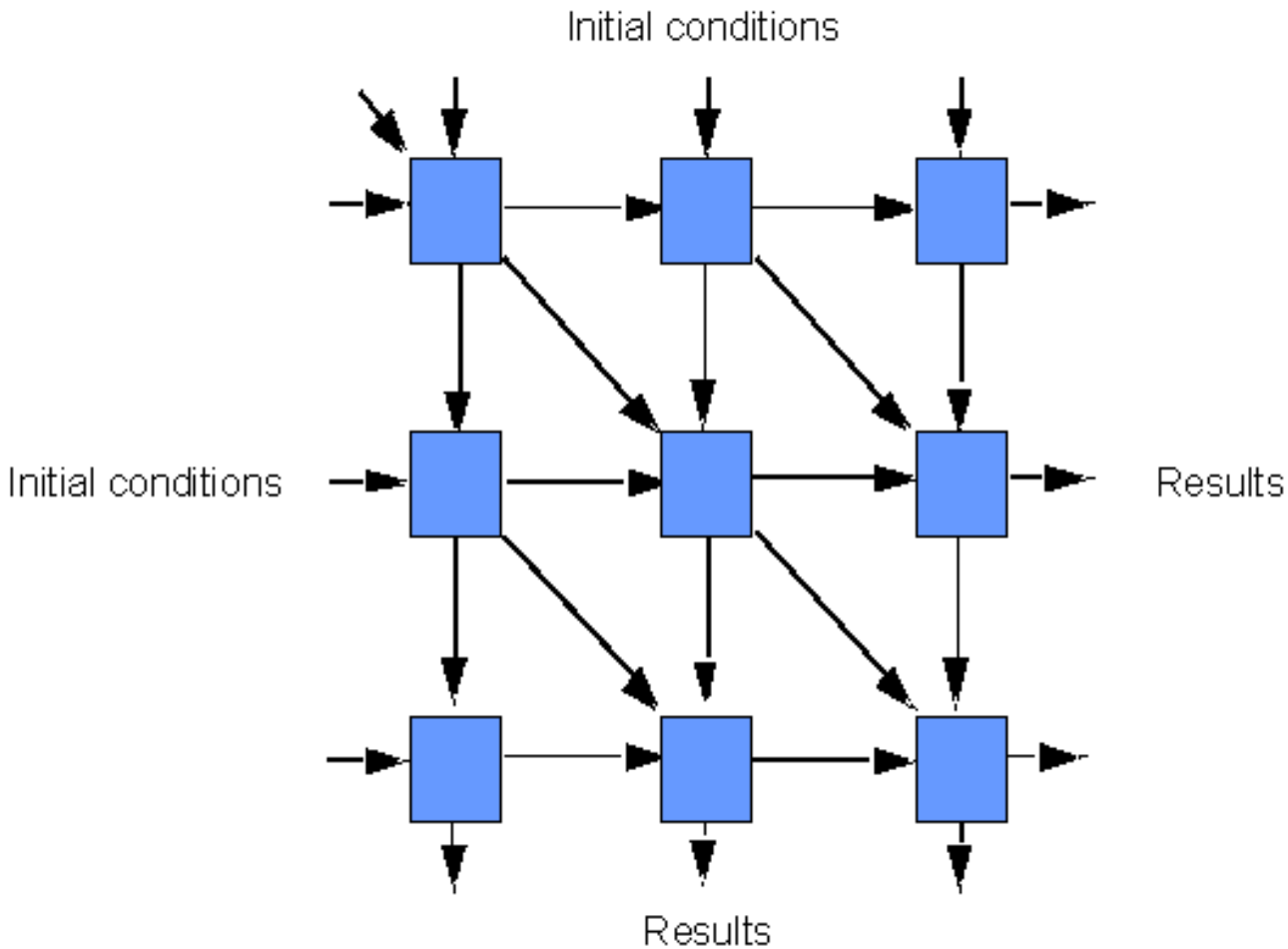
Application to the comparison of sequences

- For one cell, the data fluxes and dependencies can be summarised as following :



Bi-Dimensional Architecture

- The first and immediate solution is to use one processor by cell:



Bi-Dimensional Architecture

- The algorithm is the following:

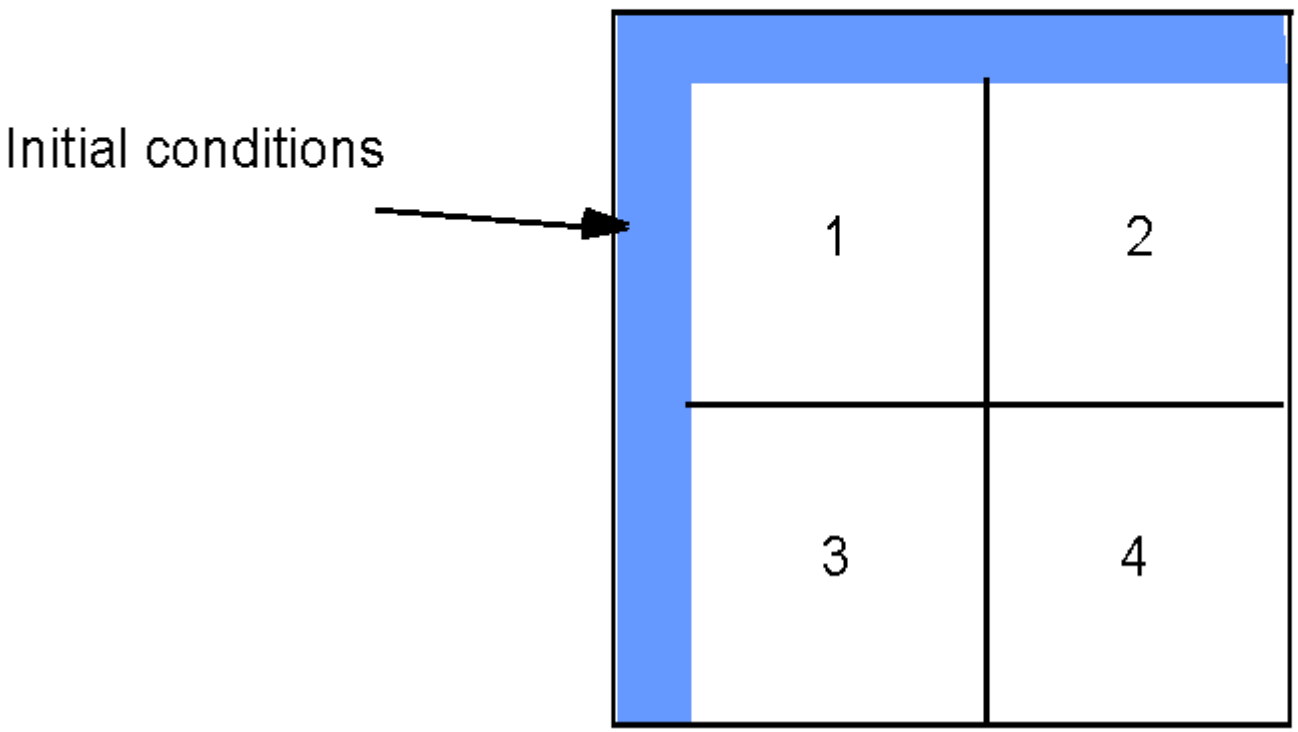
Wait for the values V_l, V_t, V_{tl}, a^i et b^i from the neighbour cells
Compute LeftVal = $V_l + W_{supp}$
Compute TopVal = $V_t + W_{ins}$
If $a^i=b^i$ **Then** DiagVal = V_{tl} Then DiagVal = $V_{tl} + W_{sub}$
Compute CellVal = **Min**(LeftVal,DiagVal,TopVal)
Send CellVal, a^i et b^i to the receiving neighbour cells

Bi-dimensional architecture

- **Advantage :**
  - The processor is simple and its size is reduced
- **Disadvantages :**
  - The quantity of processors required is equal to m.n. Since we want to be able to process sequences of more than tenth of thousand characters, the number of processors needed is so so hugely huge…pouh...
  - We need use pipelining if we want to use the processors at their full potential.
  - The interconnections to extend this architecture can be complex.

Bi-dimensional architecture

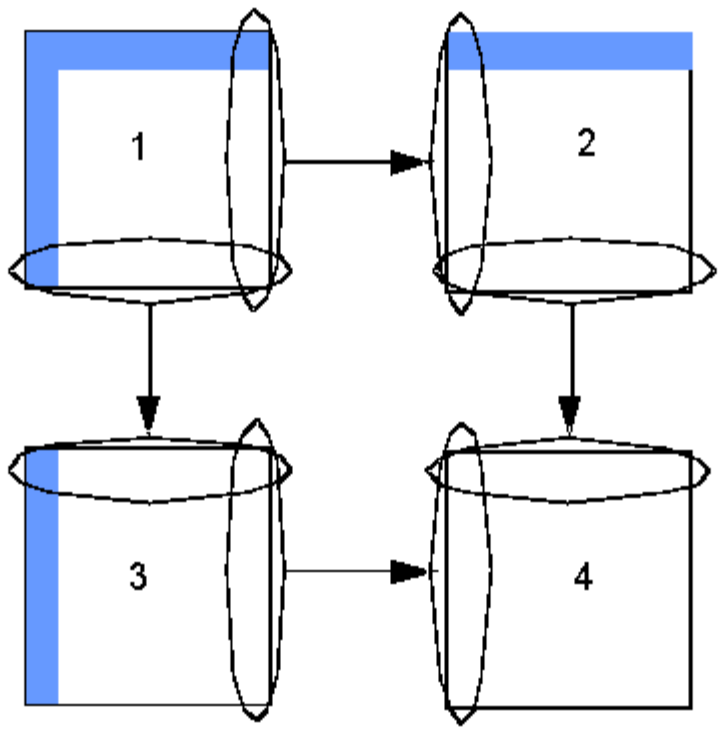
- Since we want to be able to process long sequences, we need to be able to divide the calculation into sub-problems.
- In this architecture, the sub-problems correspond to sub-matrixes.



Bi-Dimensional architecture

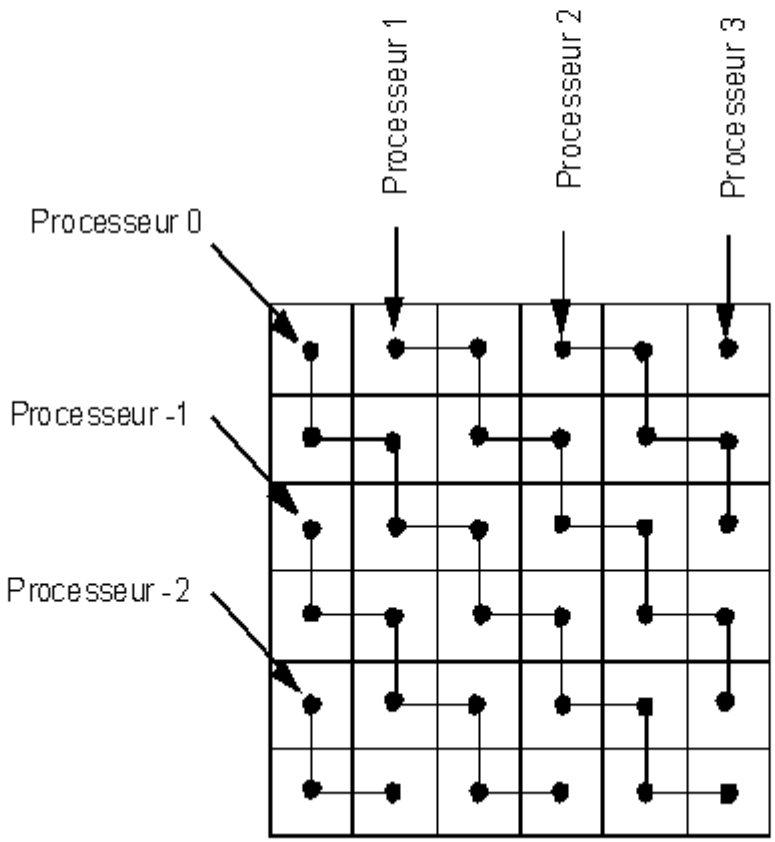
- In the evaluation of the distance, after division in sub-problems, dependencies appear : it is not possible to compute the sub-problem 4 before the sub-problems 2 and 3, for example.

The result of the sub-problem 1 is used as initial condition for the sub-problem 3.



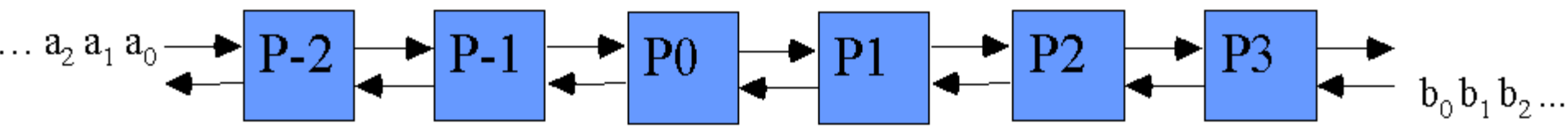
Uni-Dimensional architecture

- A solution to improve the use of the processor is to associate several cells to them.
- As a result of the data dependencies, the solution below can be proposed :



Uni-Dimensional architecture

- We can see that a processor evaluate one cell at a time.
- The input data are coming alternatively from the top cell or the left cell.
- The resulting architecture is the following :

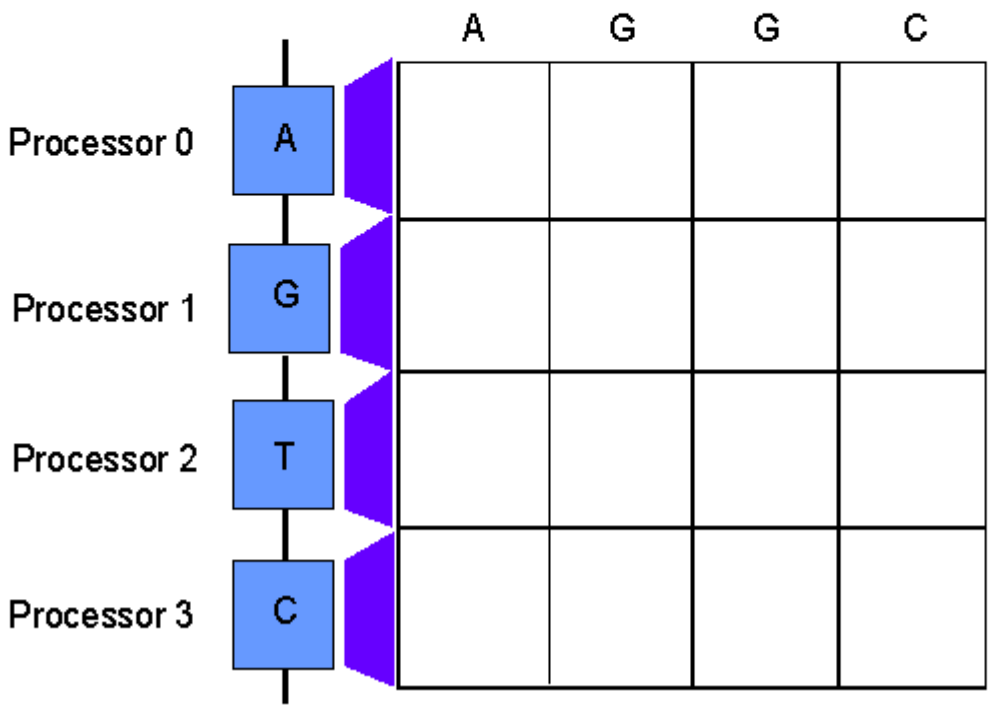


Uni-dimensional architecture

- **Advantages :**
  - The number of processor is proportional to the length of sequences.
  - The interconnection between processors is simple.
- **Disadvantages :**
  - The processors are more complex than in the bi-dimensional architecture.
  - It is not easy to divide large problems.

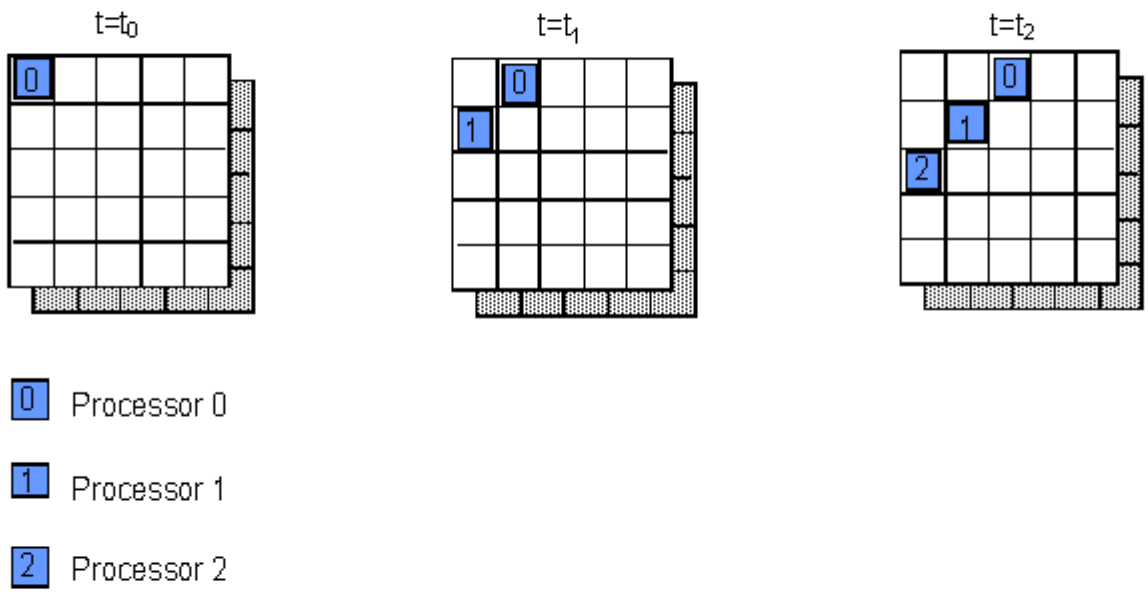
Uni-dimensional architecture

- A more satisfying solution is to associate one processor by line of the matrix.
- The result is as following :



Line uni-dimensional architecture

- Evaluation timing in the line uni-dimensional architecture :

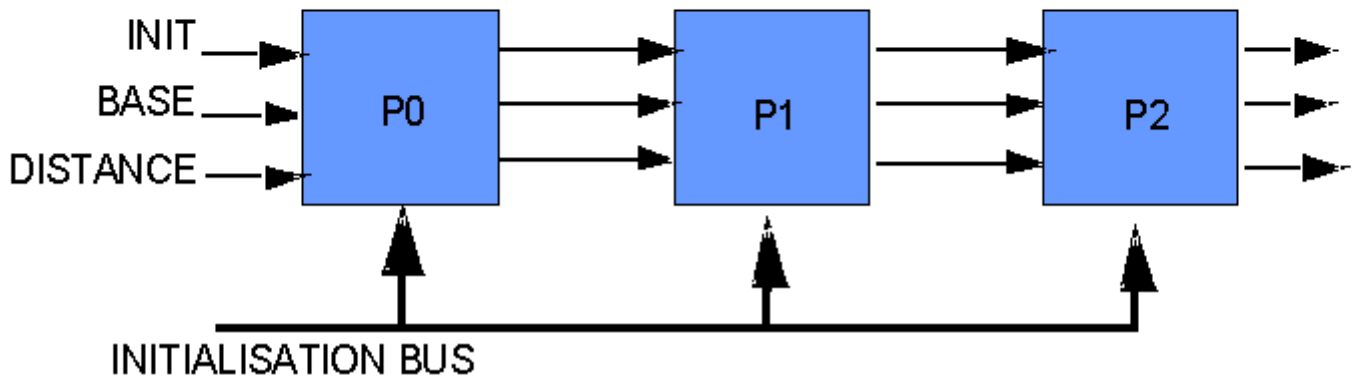


Line uni-dimensional architecture

- Each base a_i is loaded only once in the processor P_i.
- Only one processor is initialised at a time.
- The bases bj cross the processors.
- The processor P_i receives V_t from the processor P_i+1
- The processor P_i contains internally the value V_l
- V_{lt} is simply the value V_t delayed one cycle.

Line uni-dimensional architecture

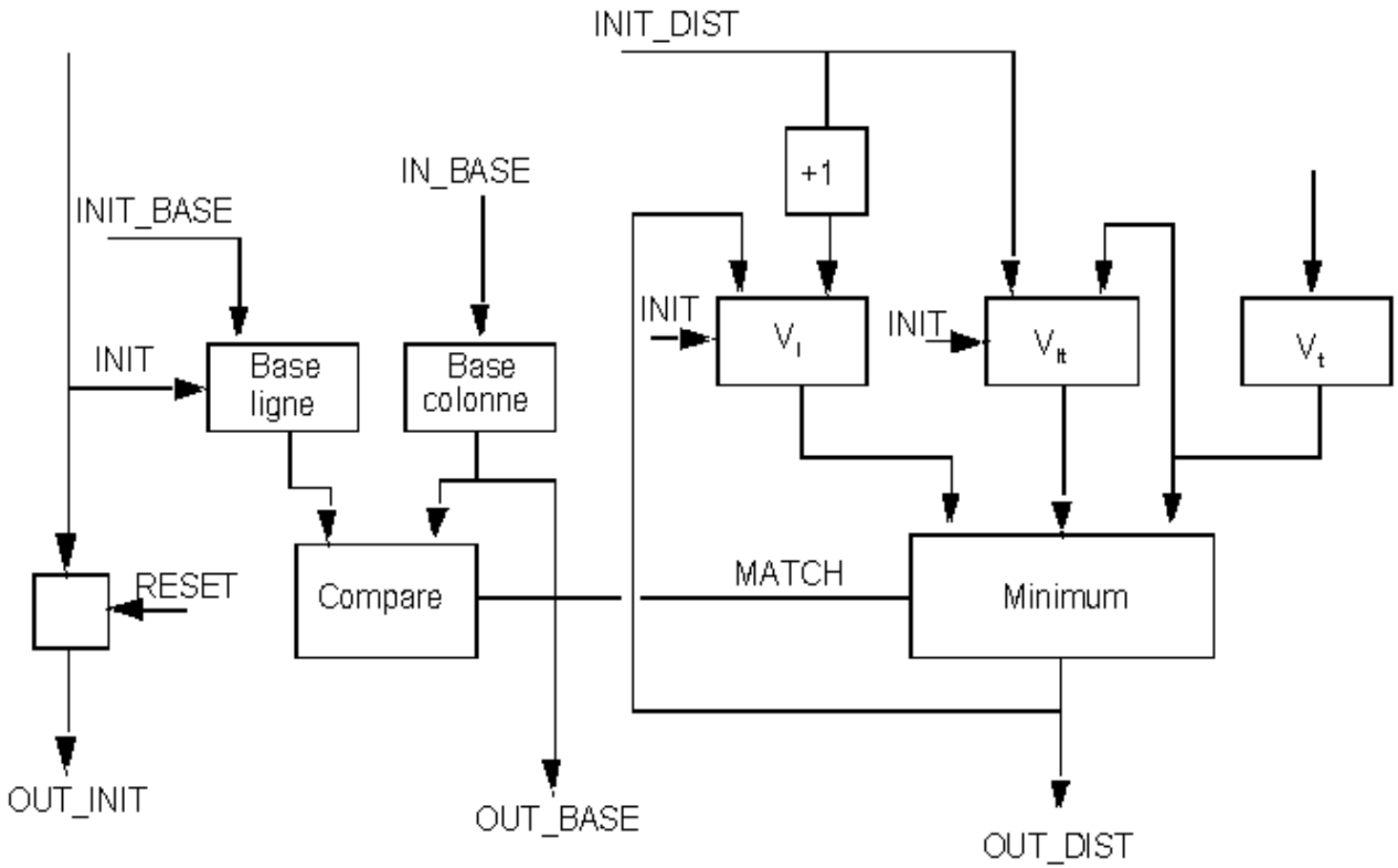
- The resulting architecture is the following:



- The initialisation bus is intended to load the a_i and initial distance values.
- The INIT bit is intended to signal the processor when to initialise itself.
- The BASE and DISTANCE signals transport respectively the b_j bases and the result of the local distance evaluation.

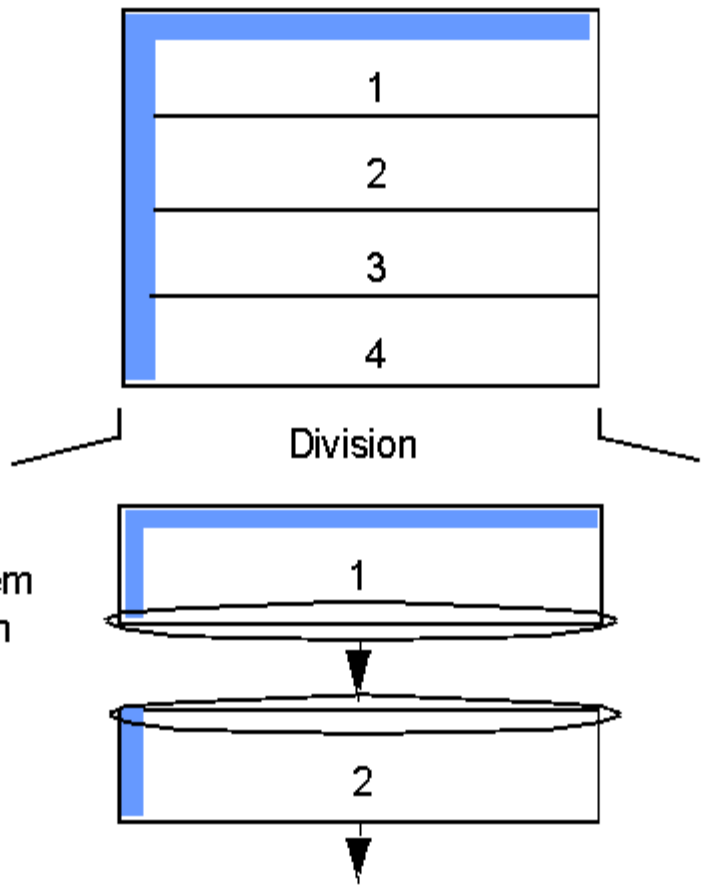
Line uni-dimensional architecture

- One processor can be detailed as following :



Line uni-dimensional architecture

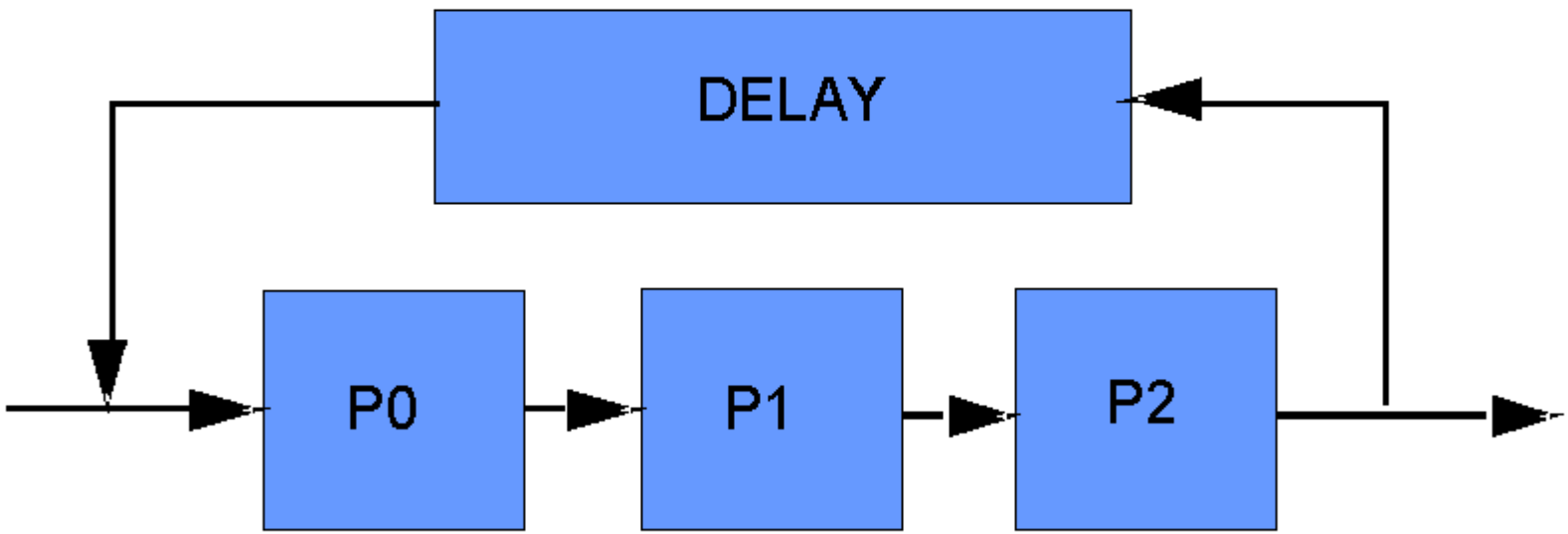
- Now we easily can divide large problems :



The output of the sub-problem i are used as initial condition of the sub-problem I+1

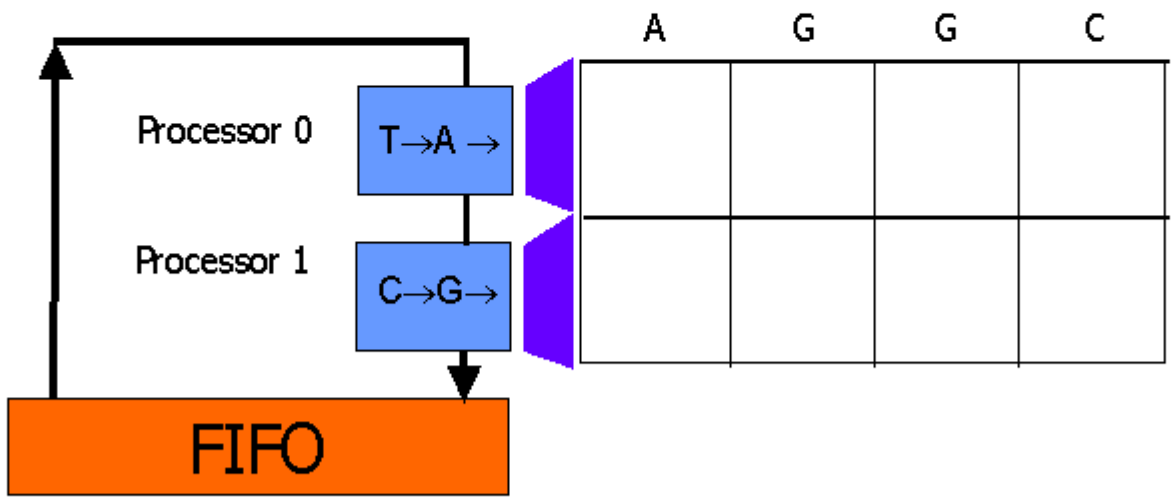
Line uni-dimensional architecture

- The division process can be simply implemented as a delay, i.e. a FIFO.



Line uni-dimensional architecture

- If we suppose we have a 4 by 4 problem and only 2 processors to resolve it, the solution is to divide the problem in two parts.
- If we take the sequences AGTC and AGGC as example, we obtain the following figure :



Line uni-dimensional architecture

- **Advantages :**
  - The number of processors is proportional to the length of sequences
  - The interconnection scheme is simple.
  - The data-flow is unidirectional.
  - It is easy to divide large problem in sub-problem.
  - The processors remain simple.
- **Disadvantage : ?**
- **This architecture has been chosen for the GENSTORM machine**

Sources

- S Cadambi, J Weener, S Goldstein,
- H Schmit and D Thomas
- Carnegie Mellon University
- Emeka Mosanya
- Swiss Federal Institute of Technology
- David E Culler, UC Berkeley

Algorithm complexity

- To compute the alignment between two sequences we also need to calculate each cell of the matrix :
  - Temporal complexity: O(n.m)
- In that case, for each cell, we also need to memorise the decision we made during the evaluation of the minimum :
  - Spatial complexity : O(n.m)