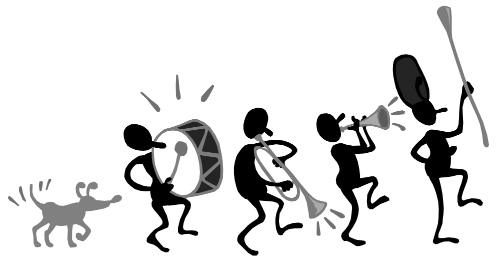
Computational models of biological systems



Giancarlo Mauri

Università di Milano-Bicocca

Complexity in biology

- **Molecular level**
  - Regulatory gene networks
  - Protein folding
- **Cellular level**
  - Cell physiology
- **Organism level**
  - Immune system
  - Nervous system
- **Population level**
  - Population dynamics
  - Ecological systems

Does Neural Communication Grow on Trees?

Analysis of interspike intervals sequences to learn and generalize correlations among neurons

The Goals

- **To search for discriminating parameters between neural substrates sottending different perceptive states**
- **To develop analysis strategies applicable to spontaneous neural activities**
- **To understand neural code**
- **To infer (thalamocortical) networks of neurons from simultaneous record of their firing activity**
- **To study the neurophysiology of (cronic) pain**

State of the art

- **Gerstein, Aertsen 1985: Crosscorrelograms to study cooperative firing activity in simultaneously recorded populations of neurons**
- **Knierim, McNaughton 2001: analysis of records of hippocampal place-cells firing through embedding in a vector space**
- **Victor, Purpura 2001: metric space based on edit distance**

State of the art

- **Rieke et al. 1997; Borst, Theunissen 1999; Johnson et al 2001: Information theoretical analysis of neural coding**

- **Panzeri et al. 1999: study of the capacity of neural channels**

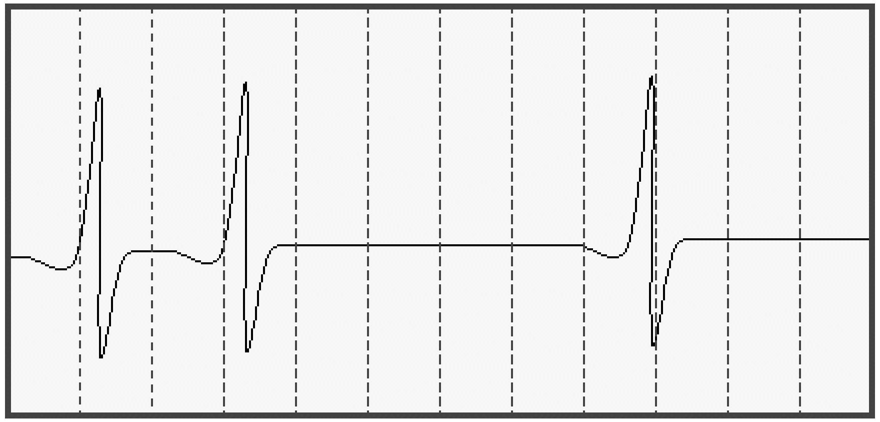
The tools

- **Longest Common Subsequence**
- **Lempel-Ziv complexity and LZ-Trees**
- **Tree Compression**

Encoding neuron's activity

Time Diagram

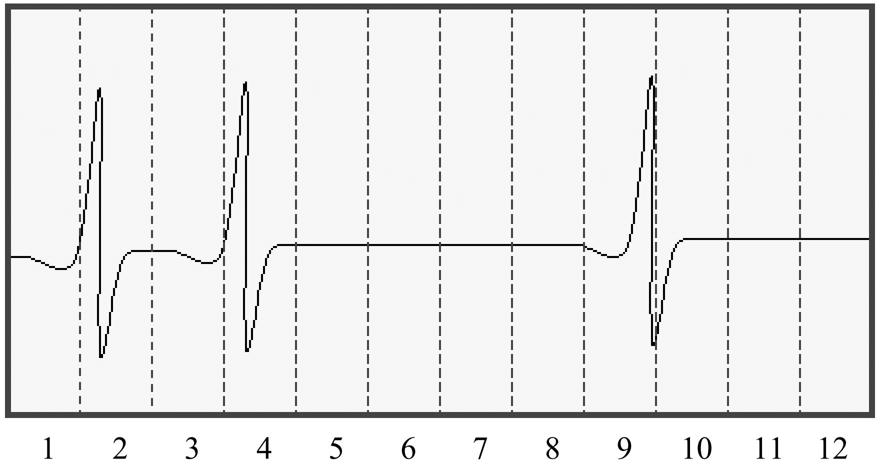
Record



Encoding neuron’s activity

Time discretization

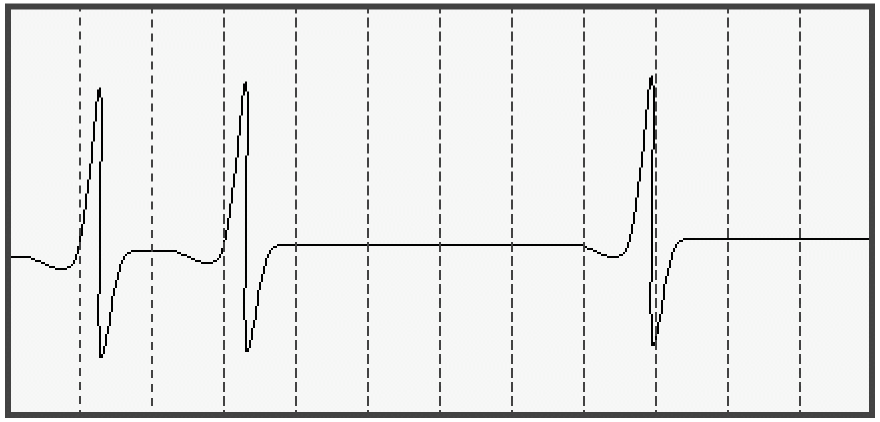
Record



Encoding neuron’s activity

Binary encoding

Record

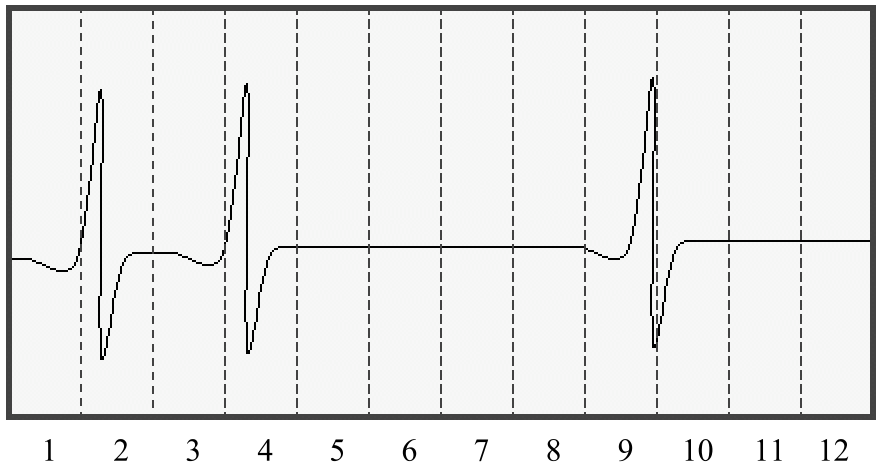


1	2	3	4	5	6	7	8	9	10	11	12
0	1	0	1	0	0	0	0	1	0	0	0

Encoding neuron’s activity

Encoding through interspike intervals

Record



Spike Times

2	4	9

Interspike Intervals

1	1	4	3

Alphabets, words, languages

Alphabet

=

finite set □ of elements called *letters,characters or symbols*

Examples

□ = {0,1}

□ = {a, b, c, ..., v, z}

□ = {A, C, G, T}

□ = {**GLY, ALA, VAL, LEU**}

Alphabets, words, languages

Word, string or sequence over □

=

function w from {1,... ,n} to □

- We write w = a_1 a_2 ... a_n where a_i = w(i) □ □
- n is the length of the sequence, denoted by |w|
- □* denotes the set of words over □

EX:	**w = AATGCA**	**|w| = 6**
	Empty word □	**|□ = 0**

Alphabets, words, languages

Concatenation of w and v, wv

=

word consisting of the characters from w, followed by the characters from v

- **ES: w = AATGCATAGGC**

  v = GGCTACT

  w v = AATGCATAGGCGGCTACT

Alphabets, words, languages

Prefix of w

=

string v such that w = vt for some t $\in \Sigma^*$

Suffix of w

=

string v such that w = tv for some t $\in \Sigma^*$

Longest Common Subsequence

Let S_1 and S_2 be two sequences over □.

S_2 is a **<u>subsequence</u>** of S_1 if it can be obtained from S_1 by removing some of its symbols

$$\begin{array}{lccccccccccccc}
S_1 = & \mathbf{T} & \mathbf{A} & \mathbf{T} & \mathbf{A} & \mathbf{G} & \mathbf{C} & \mathbf{G} & \mathbf{C} & \mathbf{A} & \mathbf{A} & \mathbf{T} & \mathbf{C} & \mathbf{G} \\
S_2 = & \mathbf{T} & \mathbf{A} & \mathbf{T} & & \mathbf{G} & \mathbf{C} & & & \mathbf{A} & & \mathbf{T} & & \mathbf{G}
\end{array}$$

S_2 is subsequence of S_1

Longest Common Subsequence

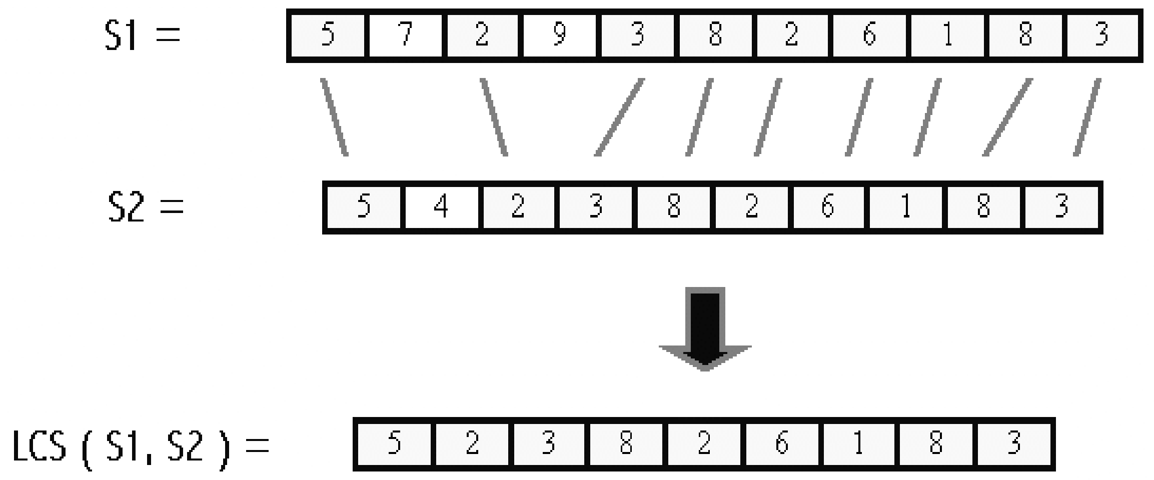
Let $\mathcal{S}$ be a set of sequences.

S is a ***common subsequence*** of $\mathcal{S}$ if it is a subsequence of every sequence in $\mathcal{S}$

<u>Problem (LCS):</u>

Given a set $\mathcal{S}$ of sequences, compute a longest common subsequence lcs($\mathcal{S}$)

Longest Common Subsequence, an example



Longest Common Subsequence

Def: **Given an alphabet □ and sequences S_1, S_2 □ □*,** lcs(**S_1, S_2) is a sequence W such that:**

1) □i, 1□ i □ |W|-1,

□j, j': 1 □ j < j' □ | S_1|, □ k, k': 1 □ k < k' □ | S_2| <u>such that</u>:

W[i]= S_1[j]= S_2[k],

<u>and</u>

W[i+1]= S_1[j']= S_2[k'];

2) ¬ □ W' □□*: (1) <u>and</u> |W'| > |W|.

LCS in sequence analysis

The lcs is able to:

- **Measure the similarity among a set of sequences through its length**
- **Exhibit the nature of the similarity through the symbols it contains**

Applications in:

- **data compression**
- **syntactic pattern recognition**
- **file comparison**
- **bioinformatics**

Complexity of LCS

- **Many polynomial time algorithms for LCS on two sequences**
- **Maier 78: LCS among k sequences is NP-hard**
- **Jiang, Li 95: nonapproximability results**
- **Jiang, Li 95: Long Run, approximation algorithm over a fixed alphabet**
- **Bonizzoni, Della Vedova, Mauri 98:better approximation ratio on the average**

LCS, Relaxed

Def: **Given an alphabet $\square$, $\square\square\square$, sequences S_1, S_2 $\square$ $\square$*, $\square$** ≥ 0,

LCS$_{\square}$(**S_1, S_2**) **is a sequence W such that:** $\square$

1) **$\square$i, 1$\square$ i $\square$ |W|-1,**

$\square$j, j': 1 $\square$ j < j' $\square$ | S_1|, $\square$ k, k': 1 $\square$ k < k' $\square$ | S_2| such that:

$$W[i] = S_1[j] = S_2[k] \pm \square,$$

and

$$W[i+1] = S_1[j'] = S_2[k'] \pm \square,$$

with $0 \square \square\square \square$;

2) $\neg$ $\square$ W'$\square\square$\*: (1) and $\square$($M_{W'}$, **S_1, S_2**) > $\square$(M_W, **S_1, S_2**),

where:

LCS, Relaxed

□S_1, S_2, W□□*,

$M_W(S_1, S_2)$:={(j, k) | 1□j□| S_1|, 1□ k□| S_2|, □i: 1□ i□|W| <u>st</u>:

W[i]= S_1[j]= S_2[k] ± □, **with** 0 □ □ □ □;

<u>and</u>

<u>if</u> 1 □ i □ |W|-1,

<u>then</u> □j’: 1□ j’□| S_1|, □k’: 1□k’□| S_2| <u>such that</u>:

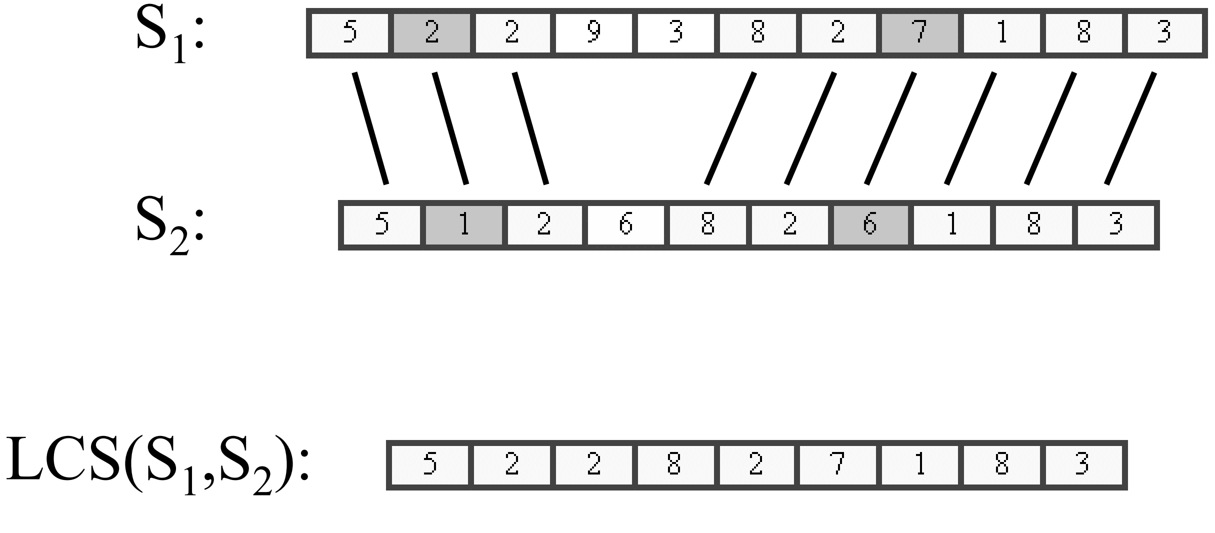
(W[i+1]= S_1[j’]= S_2[k’] ± □) □ (j’>j) □ (k’>k),

with 0 □ □ □ □; **}**

<u>and</u> where: □(M, S_1, S_2):= $_{(j, k) □ M}$cost(S[j], S[k]);

<u>and</u> cost(a, b):=1-|a-b|, with a, b □□.

LCS (Relaxed), an example



Lempel-Ziv complexity

- **L. & Z. propose as a complexity measure of a sequence the minimum number of steps needed to produce it from its prefixes using copy and paste operations**

- **L. & Z. give an algorithm to compute the above measure**

- **The complexity notion defined by L. & Z. is compatible with the algorithmic complexity theory (Kolmogorov, Chaitin)**

Lempel-Ziv Algorithm

INPUT: $S \in \Sigma^*$; OUTPUT: $w = \{Q \in \Sigma^* \mid \exists i, j: S[i:j]=Q\}$;

```
w := $\emptyset$;
w := w $\cup$ {$\varepsilon$};
curr := 1;
while curr $\leq$ |S| do
      begin
         S' := S[curr:n] s.t. S' $\in$ w and S'°S[n+1] $\notin$ w;
         w := w $\cup$ {S'°S[n+1] };
         curr := n+2;
      end
```

<u>NOTE:</u> $S[i:j]=\varepsilon$ for $j<i$

Lempel-Ziv -Trees

- **The vocabulary** w **obtained can be organized in a hierarchical (tree) structure through the prefix relation:**

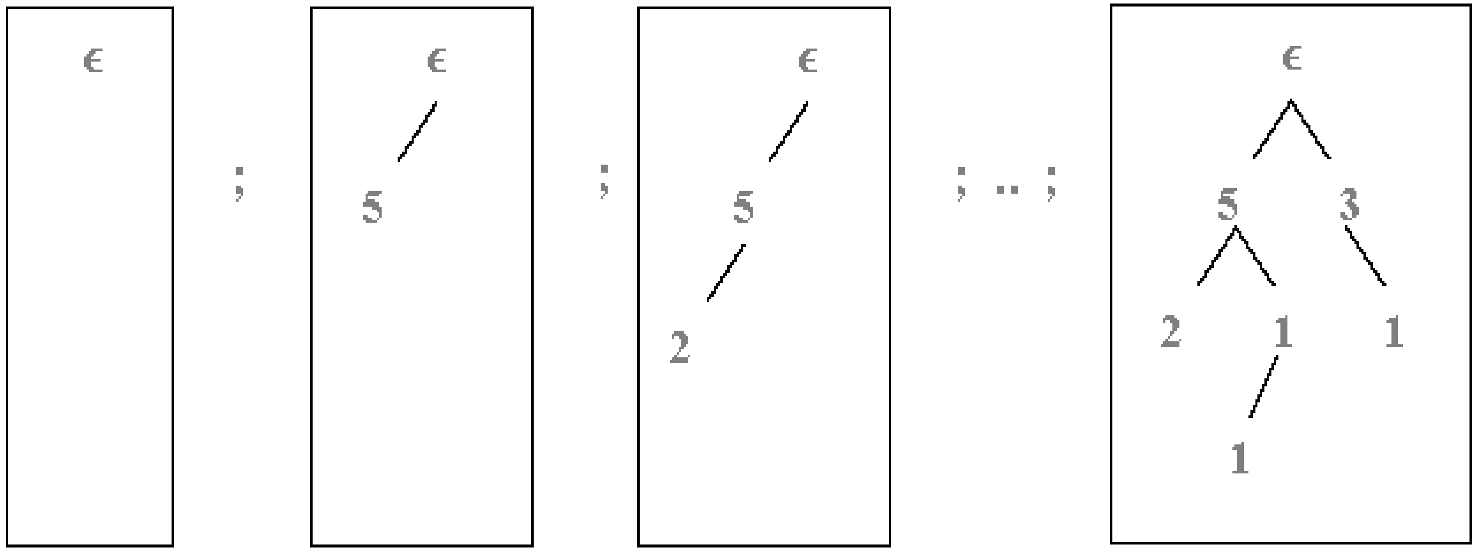
prefix := { (u, v) | u, v $\in$ w and $\exists$i: u=v[1:i] };

- **Every word in w (except ε) can be obtained by adding a single symbol to another word in w; hence, it can be encoded through a pointer to its maximal prefix, plus the last symbol**

- **LZCompl(S) := |w| / |S|**

Lempel-Ziv-Trees, an example

$S = 5 \bullet 52 \bullet 3 \bullet 51 \bullet 31 \bullet 511 \bullet$

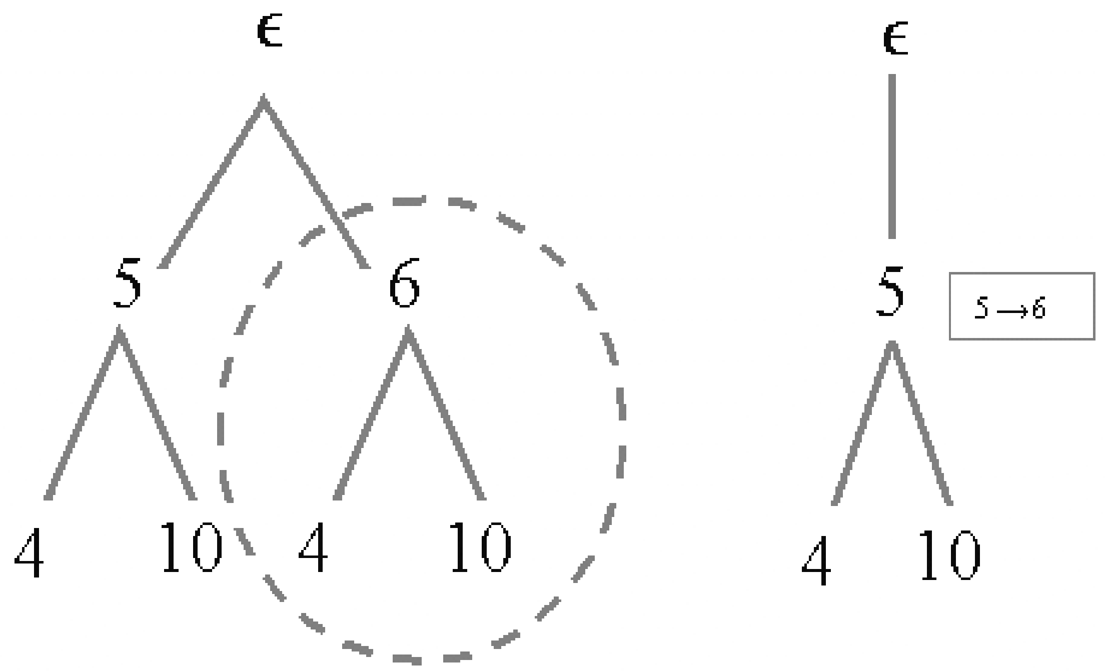


Lempel-Ziv-Trees, meaning

- **Acquisition of knowledge about the regularity of occurrence of symbol patterns in the sequence**

- **Structuring of knowledge so as to give a representation of the sequence shortest than the list of its symbols.**

Tree Compression, an example



Tree Compression, meaning

- **Reduction of redundancy in the tree structure**
- **Minimization of hierarchical knowledge representations**
- **Abstraction and generalization of the knowledge empirically acquired**

Edit Distance between trees

Let T be a rooted labeled tree over a given alphabet □ :

T = < V, E, r, lab: V□ □ >

and let have the following operations on it :

- Insertion **of an element: □□ a, a□□;**
- Deletion **of an element: a□ □, a□□;**
- Substitution **of the label of an element: a□ b, a, b □□;**

Edit Distance between trees

EditOps := $\{a \rightarrow b \mid a, b \in \Sigma \cup \{\lambda\}\} \setminus \{\lambda \rightarrow \lambda\}$;

Given the (metric) cost function :

$$\gamma : \text{EditOps} \rightarrow R^{+};$$

We define the cost of a sequence Sop $\in$ EditOps* as

$$\gamma(\text{Sop}) = \sum_{i=1,..,|\text{Sop}|} \gamma(\text{Sop}[i]).$$

Edit Distance between trees

Def: **Given two labeled trees T e T’, the edit distance between them is defined by:**

Edist(T, T’) :=

$$\min_{Sop \square EditOps^*} \{ \square(Sop) \mid T' = Sop(T) \}.$$

Tree Compression, Algorithm

```
proc TreeCompr( tot $\in$ R, < &T, &Sop > ) :

if ( $V_T \neq \emptyset$ ){
        if ( Edist(Tdx($r_T$), Tsx($r_T$)) < threshold ) {
            Prune(Tdx($r_T$));
            TreeCompr( tot, < Tdx, Sop°$Sop_{Edist(Tdx(rT), Tsx(rT))}$ > );
        } else {
            TreeCompr( tot, < Tdx, Sop > );
            TreeCompr( tot, < Tsx, Sop > );
          }
}
```

Tree Complexity

Def: **given a tree T, let T’ and Sop□EditOps the results of the compression of T through** TreeCompr**; the Tree Complexity of T is:**

TC(T) := (|T’| / |T|)

+□·□(Sop)

where 0 □□ □1

Tree Complexity

Teorema: **The computation of the tree complexity ofa tree T based on an Edit Distance *Structure Respecting* has time complexity** :

$$O(D^3 \cdot |T|^2),$$

where D is the maximum degree of nodes in T.

Application

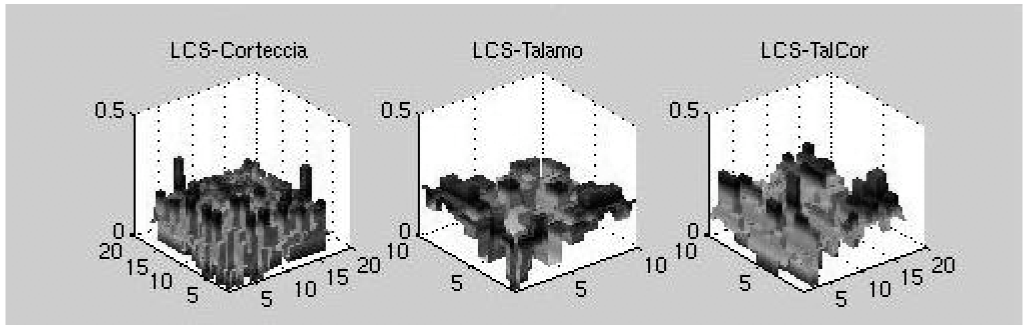
Analysis of sequences of *Interspike Intervals* from simultaneous recordings of talamic and cortical cells populations.

Motivation: **key role of talamocortical areas in the elaboration of somatosensorial stimuli.**

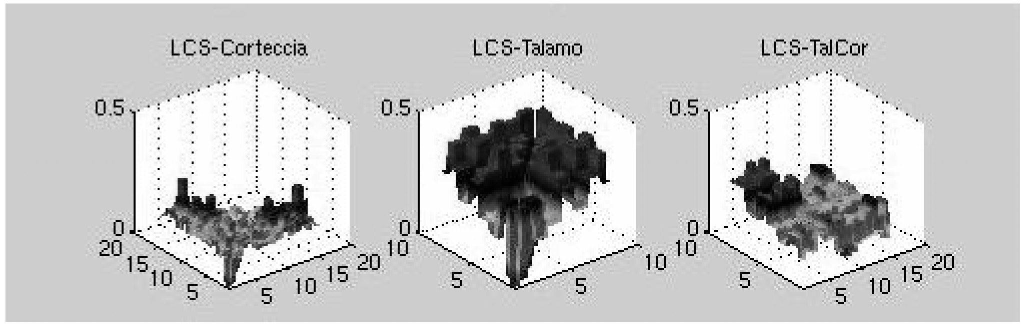
Goal: **to discover rythmic correlations among cells activities.**

Application, LCS

NORM:

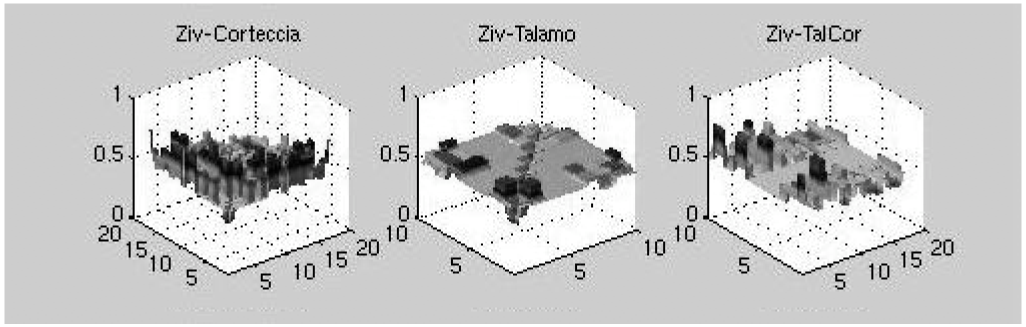


CCI:

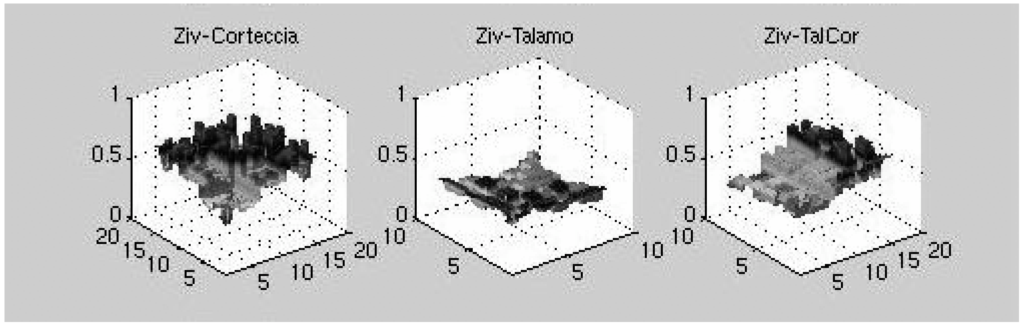


Application, LZ-Complexity

NORM:

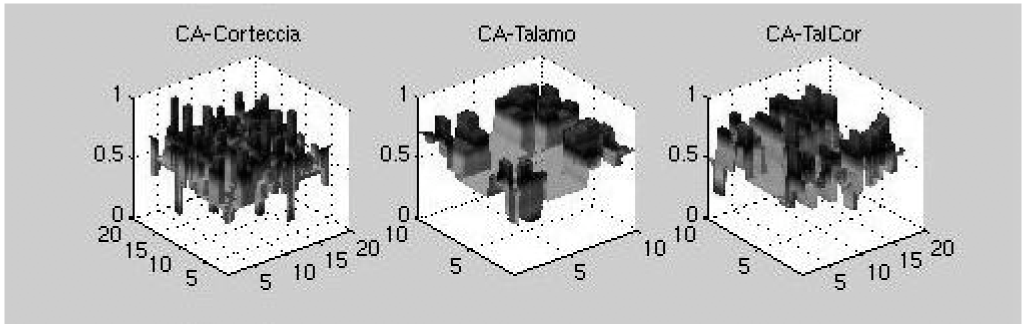


CCI:

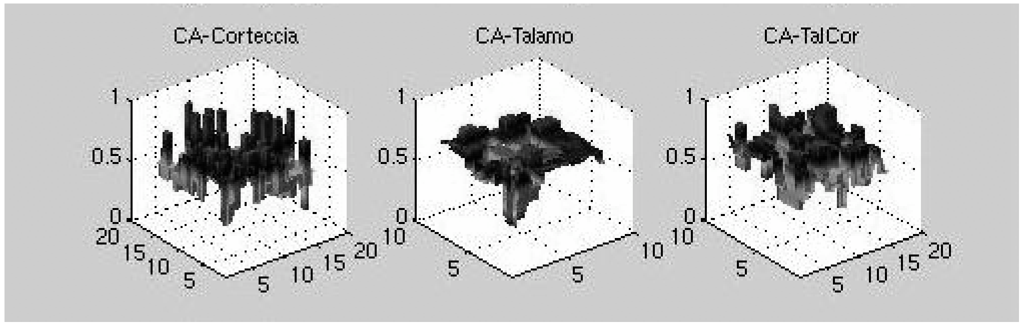


Applicazione, CplArb

NORM:



CCI:



Application, conclusions

The three kinds of di analysis help us to enlightening different aspects of the process we are observing:

• LCS	**Omogeneity**
• Ziv-Tree	**Monotonicity**
• Tree compression	**Fault Tolerance**