Networks of Genetic Processors

José M. Sempere
Research Group on Computation Models and Formal Languages
Departmento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Networks of Genetic Processors

1. Previous Models: NEPs and NSPs
2. Accepting NGPs
3. NGPs are computationally complete
4. Some considerations on computational and description complexity
5. Other variants of NGPs: Generators and Optimizers
6. Parallel Genetic Algorithms and NGPs
7. Solving the Optimization Traveling Salesman Problem
8. Work in progress: Applying NGPs to solve multialignment in DNA/protein sequences

Networks of Genetic Processors

Some bioinspired operators over strings and languages

Insertion Insert a symbol into a string — aaaaa ➔ aabaaa

Deletion Delete a symbol from a string — aabaaa ➔ aaaaa

Substitution (mutation) Substitute a symbol into a string — aaaaa ➔ aabaa

Splicing Splicing rules $r=(u_1\#u_2\$v_1\#v_2)$ — r=(a#a$b#b) (abcdaa,bbabcd) ➔ (abcdababcd,ba)

Crossover Full massive splicing with empty context — aa ⊳⊲bb ➔ λ, bb, abb, aabb, ab, aab, …

Hairpin completion Hairpin completion from folded strings

Superposition Complementarity completion from double stranded strings

loop and double loop recombination DNA recombination based on gene assembly

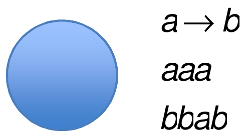
inversion, duplication and transposition DNA fragments modification (operations on substrings)

… etc, etc.

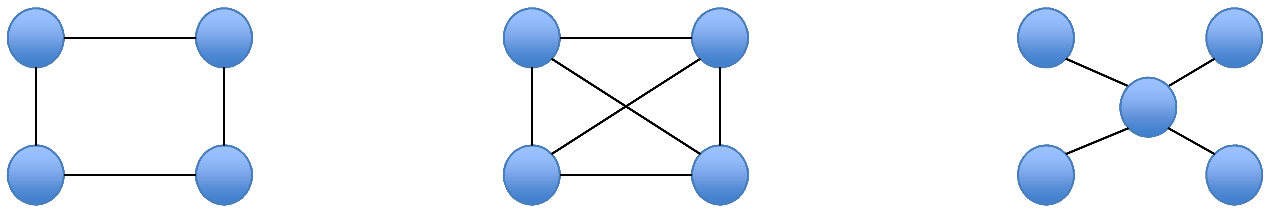
Networks of Genetic Processors

The ingredients to define a Network of Bioinspired Processors

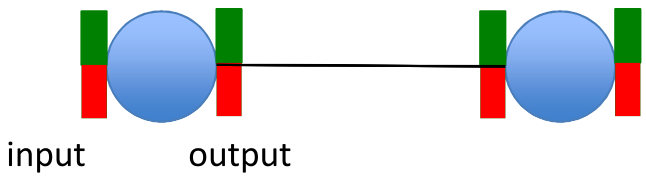
A finite set of processors that apply operations over strings which have been inspired by biomolecular functions and operations in the nature. The processors work with a multiset of strings.

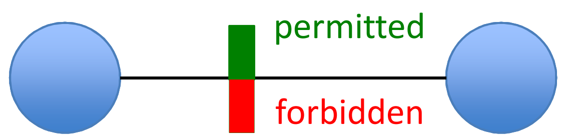


A connection topology between processors in the form of a network.



A set of (input/output) filters which can be attached to the processors or to the connections.





Networks of Genetic Processors

Accepting Networks of Evolutionary Processors (I)

An evolutionary processor over *V* is a 5-tuple *(M,PI,FI,PO,FO)*,where:

Either $M \subseteq Sub_V$ or $M \subseteq Del_V$ or $M \subseteq Ins_V$
The set M represents the set of evolutionary rules of the processor.

$PI,FI \subseteq V$ are the input permitting/forbidding contexts of the processor

$PO,FO \subseteq V$ are the output permitting/forbidding contexts of the processor
(with $PI \cap FI = \emptyset$ and $PO \cap FO = \emptyset$)

We can define the following predicated for the filters

$$rc_s(z;P,F) \equiv [P \subseteq alph(z)] \wedge [F \cap alph(z) = \emptyset]$$

$$rc_w(z;P,F) \equiv [alph(z) \cap P \neq \emptyset] \wedge [F \cap alph(z) = \emptyset]$$

Networks of Genetic Processors

Accepting Networks of Evolutionary Processors (II)

$$\Gamma = (V, U, G, N, \alpha, \beta, x_I, x_O)$$

where

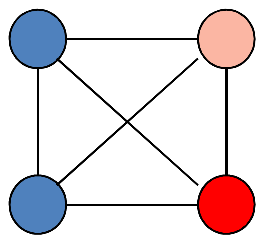
V and U are the input and network alphabets
$G=(X_G,E_G)$ is an undirected graph without loops
$N: X_G \rightarrow EP_U$ associates an evolutionary processor to every node in G
$\alpha: X_G \rightarrow \{l,r,*\}$ associates an action mode to every node (Hybrid networks)
$\beta: X_G \rightarrow \{s,w\}$ associates a filter predicate to every node
x_I,x_O are the input and output nodes

Networks of Genetic Processors

Accepting Networks of Evolutionary Processors (III)

$$\Gamma = (V, U, G, N, \alpha, \beta, x_I, x_O)$$

How does the network work ?



(I) Evolutionary steps

$$C_i \Rightarrow C_{i+1}$$

- *Every rule that can be applied is massively applied*
- *No competition between rules. All the rules are applied by using different copies*

(II) Communication steps

$$C_i \mapsto C_{i+1}$$

- *Every processor sends all the filtered strings to its neighbours*
- *Every processor receives and stores filtered strings*
- *Strings that are sent but not received are lost*

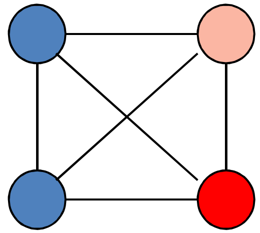
(III) Network at work

$$C_0 \Rightarrow C_1 \mapsto C_2 \Rightarrow C_3 \mapsto C_4 \ldots$$

Networks of Genetic Processors

Accepting Networks of Evolutionary Processors (IV)

$$\Gamma = (V, U, G, N, \alpha, \beta, x_I, x_O)$$



Accepted language

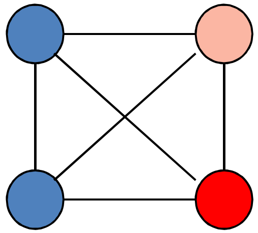
1. There exists a configuration in which the set of words existing in the output node x_O is non-empty. (halting and accepting computation)
2. There exist two consecutive identical configurations. (halting and rejection computation)
3. It works forever.

$L(\Gamma)=\{w \in V^*$: the computation of Γ on w is an accepting one$\}$.

Networks of Genetic Processors

Accepting Networks of Splicing Processors

$$\Gamma = (V, U, G, N, \alpha, x_I, x_O)$$



Let $r=u_1\#u_2\$v_1\#v_2$ an splicing rule then $(x,y) \mapsto_r (w,z)$ iff $x=x_1u_1u_2x_2$, $y=y_1v_1v_2y_2$, $w=x_1u_1v_2y_2$ and $z=y_1v_1u_2x_2$

$$\sigma_R(L) = \{z, w \in V^* : (\exists u, v \in L, r \in R)[(u,v) \mapsto_r (z,w)]\}$$

An splicing processor over *V* is a 5-tuple *(S,A,PI,FI,PO,FO)*,where:

S is a finite set of splicing rules
A is a finite set of auxiliary words
The rest of elements are identical to the evolutionary case

Splicing steps $C'(x) = \sigma_{S_x}(C(x) \cup A_x)$

SAME ACCEPTANCE CRITERION AS IN THE EVOLUTIONARY CASE

Networks of Genetic Processors

From ANEPs and ANSPs to Accepting Networks of Genetic Processors (ANGPs)

Substitute evolutionary operations or splicing rules by

(a) Mutation operations

$a \rightarrow b$

aaaaa

baaaa
abaaa
aabaa
aaaba
aaaab

(b) Crossover between strings

$$x \bowtie y = \{ x_1y_2, y_1x_2 : x=x_1x_2, y = y_1y_2 \}$$

	cd	cd	cd
ab	cd,ab	d,cab	λ,cdab
ab	acd, b	ad,cb	λ,cdb
ab	abcd, λ	abd,c	ab,cd

Networks of Genetic Processors

From ANEPs and ANSPs to Accepting Networks of Genetic Processors (ANGPs)

Some important remarks:

(1) NEPs with only substitution (mutation) processors are not computationally complete
(2) NSPs with empty contexts (*crossover*) are not computationally complete

Combine mutation and crossover to …

(1) achieve computation completeness
(2) Connect Networks of Bio-Inspired Processors with Genetic Algorithms

Networks of Genetic Processors

Accepting Networks of Genetic Processors (I)

A genetic processor over *V* is a 5-tuple $(M_R, A, PI, FI, PO, FO, \alpha, \beta)$,where:

- MR is a finite set of mutation rules over V (a → b)
- A is a multiset of strings over V with finite support and an arbitrary large number of copies of every string
- PI,FI $\subseteq$ V* are finite sets of input permitting/forbidding contexts
- PO,FO $\subseteq$ V* are finite sets of output permitting/forbidding contexts
- $\alpha \in$ {(1),(2)} defines the function mode such that

  (1) means only mutation operations
  (2) means crossover operations (MR = $\varnothing$)

- $\beta \in$ {(1),(2)} defines the filter predicates

  (1) $rc_s(z, P, F) \equiv [P \subseteq seg(z)] \wedge [F \cap seg(z) = \varnothing]$
  (2) $rc_w(z; P, F) \equiv [seg(z) \cap P \neq \varnothing] \wedge [F \cap seg(z) = \varnothing]$

Networks of Genetic Processors

Accepting Networks of Genetic Processors (II)

ANGP of size *n* is a tuple $\Gamma = (V, N_1, N_2, \ldots, N_n, G, N)$

where V is an alphabet

$G=(X_G,E_G)$ is an undirected graph without loops

N_i $(1 \le i \le n)$ is a genetic processor over V

$N: X_G \rightarrow \{N_1,N_2,\ldots,N_n\}$ associates a genetic processor to every node in the graph

How does the network work ?

(I) Genetic steps

$C_i \Rightarrow C_{i+1}$

- *Every rule that can be applied is massively applied*
- *No competition between rules. All the rules are applied by using different copies*

(II) Communication steps

$C_i \mapsto C_{i+1}$

- *Every processor sends all the filtered strings to its neighbours*
- *Every processor receives and stores filtered strings*
- *Strings that are sent but not received are lost*

(III) Network at work $C_0 \Rightarrow C_1 \mapsto C_2 \Rightarrow C_3 \mapsto C_4 \ldots$

SAME ACCEPTANCE CRITERION AS IN THE EVOLUTIONARY CASE

Networks of Genetic Processors

Theorem: ANGPs are computationally complete

From deterministic Turing machines

$$M = (\Sigma, \Gamma, Q, \delta, q_0, B, Q_f)$$

instantaneous description αqaβ

movement to the right δ(q,a)=(p,b,R)

movement to the right δ(q,a)=(p,b,L)

visit to a new rightmost cell

αqB

.... to ANGPs

$$R = (V, N_c, N_1, \ldots, N_n, N_{out}, \hat{K}, f)$$

encoded instantaneous description qα\$aβF

dedicated processor N_{qaR} $q \to p$, $\$ \to b'$, $a \to \overline{\$}$ strong PI = {q,\$a}

a couple of dedicated processors for every c

N_{qacL1} $q \to p$, $\$ \to \underline{c}$, $a \to b'$ strong PI = {q,\$a} N_{qacL2} $c \to \overline{\$}$ strong PI = {p,c$\underline{c}$b'}

a couple of dedicated processors

N_B crossover with #BF N_{B2} restores #BF

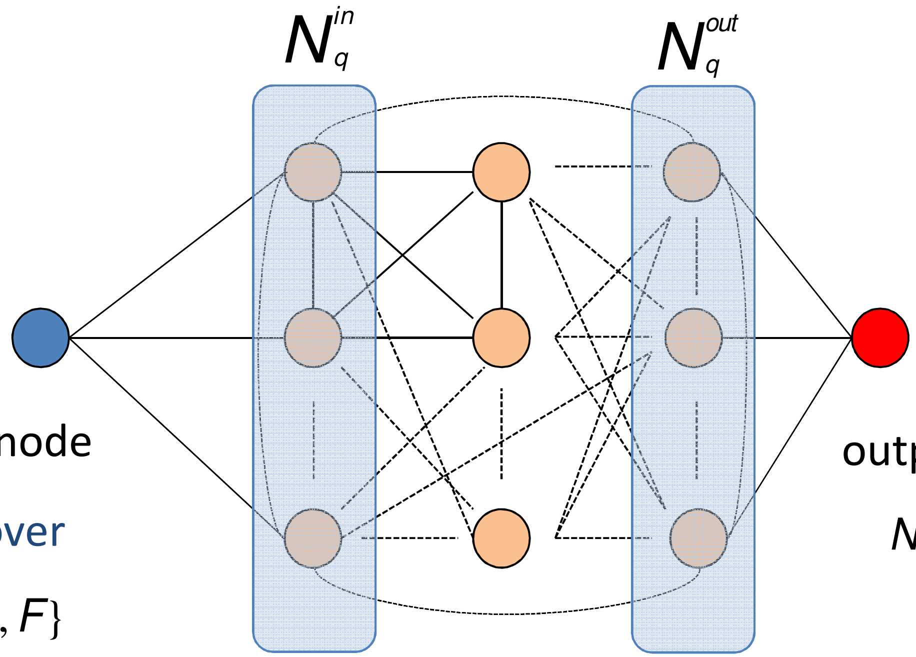
qα\$BF $\in$ qα\$F $\bowtie$ #BF

Networks of Genetic Processors

Theorem: ANGPs are computationally complete

$$\Gamma = (V, N_1, N_2, \ldots, N_n, G, N)$$

The network topology



input node

output node

crossover

$\{w, q_0 \overline{\$}, F\}$

$q_0 \overline{\$} wF$

fully connected subgraph

$\hat{K}$

$$N_{out} = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, (2), (1))$$

Networks of Genetic Processors

Theorem: ANGPs are computationally complete

A similar simulation for non-deterministic Turing machines

$$M = (\Sigma, \Gamma, Q, \delta, q_0, B, Q_f)$$

instantaneous description αqaβ

movement to the right

(p,b,R) ∈ δ(q,a)

movement to the right

(p,b,L) ∈ δ(q,a)

$$R = (V, N_c, N_1, \ldots, N_n, N_{out}, \hat{K}, f)$$

encoded instantaneous description qα$aβF

dedicated processor N_{qapbR} $\begin{array}{l} q \to p \\ \$ \to b' \\ a \to \overline{\$} \end{array}$ strong PI = {q,$a}

a couple of dedicated processors for every c

$N_{qapbcL1}$ $\begin{array}{l} q \to p \\ \$ \to \underline{c} \\ a \to b' \end{array}$ strong PI = {q,$a} $N_{qapbcL2}$ $c \to \overline{\$}$

strong PI = {p,c$\underline{c}$b'}

Networks of Genetic Processors

Looking to the computational complexity

Let us consider an ANGP R and the language L accepted by R, then the time complexity of the accepting computation of R if x is given as an input string is denoted by $Time_R(x)$ and it is defined as the number of steps (both communication and evolutionary ones) such that the network R halts on x in an acceptance mode.

$$Time_R(n) = \max\{Time_R(x) : x \in L(R), |x| = n\}$$

$Time_{ANGP}(f)$={L: There exists an ANGP, R, and a natural number n_0 such that L=L(R) and for all $n \geq n_0$, $(Time_R(n) \leq f(n))$}

$$Time_{ANGP}(C) = \bigcup_{f \in C} Time_{ANGP}(f)$$

$$Time_{ANGP}(poly) \equiv PTime_{ANGP}$$

Theorem: $NP \subset PTime_{ANGP}$

Open Problem: $PTime_{ANGP} \subset ?$

Networks of Genetic Processors

Other variants of Networks of Genetic Processors

Generating Networks of Genetic Processors (GNGPs)

No input processor
The output processor collects the generating language
The halting criterium is the repetition of two consecutives configuration

Theorem: Let L be a recursively enumerable language generated by a grammar G in Kuroda's Normal Form. Then, there exists a GNGP R such that

(1) R has 16 genetic processors

(2) R generates L

Optimizing Networks of Genetic Processors (ONGPs)

The input processor stores the instance of the problem P to be optimized according to f
The output processor collects the solution S such that, at anytime t,

$$S = \text{argmax}/\min(f,t) : (\forall t_i \leq t)(f(S_{t_i}) \leq / \geq f(S))$$

No explicit halting criteria
The processor filters can be substituted by integer functions and threshold values

Networks of Genetic Processors

From Networks of Genetic Processors to Parallel Genetic Algorithms

The main components of a Genetic Algorithm (or Evolution Program) are:

- A genetic representation for potential solutions to the problem
- A way to create an initial population of potential solutions
- An evaluation function that plays the role of the environment, rating solutions in terms of their "*fitness*"
- Genetic operators that alter the composition of the potential solutions
- Values for various parameters that the genetic algorithm uses(population size, probabilities of applying genetic operators,etc.)

Networks of Genetic Processors

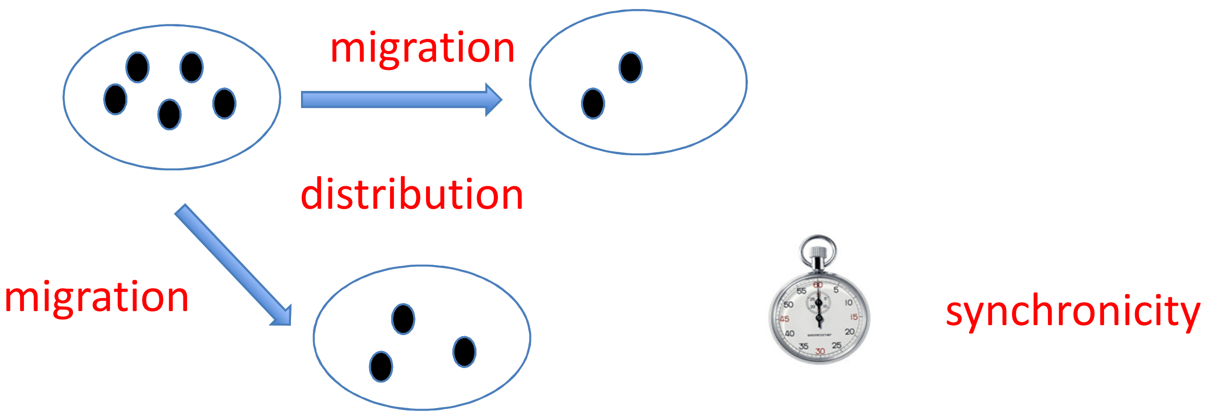
From Networks of Genetic Processors to Parallel Genetic Algorithms

The main ingredients to propose Parallel and Distributed Genetic Algorithm

The distribution of the individuals in different populations (master-slave, multiple populations or islands, fine-grained populations or hierarchical and hybrid populations) and the neighborhood topology (rings, *m,n*-complete, ladders, grids, etc.)

The synchronicity of the populations evolution and communication.

The migration phenomena: The migration rates (the percentage of individuals that migrate from one population to a different one),the migration selection (the selections of the individuals that migrate) and the migration frequency.



Networks of Genetic Processors

From Networks of Genetic Processors to Parallel Genetic Algorithms

From (Parallel) Genetic Algorithm as optimizers to acceptors

Acceptance Criterion I

Let *w* be an input string. We will say that a PGA accepts w if, after a finite number of iterations (operator applications, fitness selection and individuals migration), w appears in a predefined survival population.

Acceptance Criterion II

Let w be an input string. We will say that a PGA accepts w if, after a finite number of iterations (operator applications, fitness selection and individuals migration), a distinguished individual x_{yes} appears in a predefined survival population. We say that the PGA rejects the input string if, after a finite number of iterations (operator applications, fitness selection and individuals migration), a distinguished individual x_{not} appears in a predefined survival population or the PGA never finishes.

Theorem: Let D be a decision problem and L_D its acceptance language. D can be solved by a Parallel Genetic Algorithm with acceptance criterion I iff it can be solved with acceptance criterion II.

Networks of Genetic Processors

From Networks of Genetic Processors to Parallel Genetic Algorithms

Theorem: Parallel Genetic Algorithms with multiple-populations, synchronicity and full migration phenomena are computationally complete.

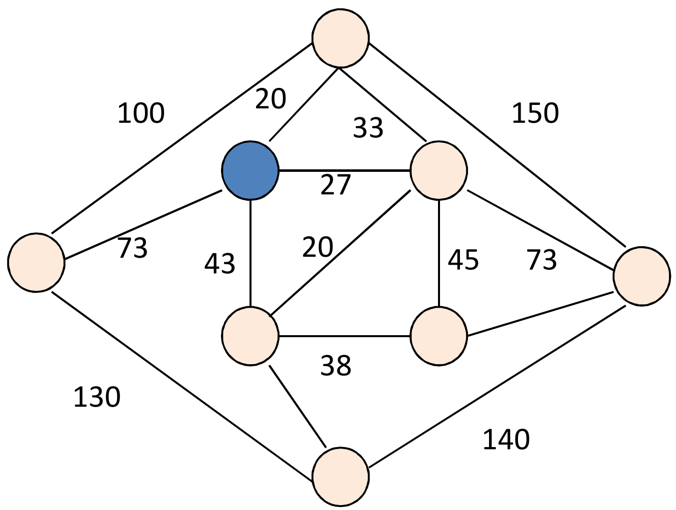
Open Problem I Is full migration phenomena really needed ?

Open Problem II What is the role of crossover ?

Networks of Genetic Processors

Solving the Optimization Traveling Salesman Problem (I)

The Problem: There are *n* cities and connections between them. We have to find a path that starts and begins at a given city, visits any city with a minimal distance



G=(N,A)

Find C={1,2,…,n} such that $C = \operatorname{argmin}\left(\sum_{i=1}^{n-1} A[C_i, C_{i+1}] + A[C_n, C_1]\right)$

Networks of Genetic Processors

Solving the Optimization Traveling Salesman Problem (II)

- The strings in the processors are the secuences of nodes in a path
- The filters at the genetic processors are replaced by fitness functions (the sum of the distances in the path) and selection of the best
- The experiments replicate *"Solving Travaling Saleman Problem by Ant Colony Optimization Algorithm with Association Rule",* G. Shang, Z. Lei, Z. Fengting, Z. Chunxian. Third International Conference on Natural Computation (ICNC 2007)
- 30 cities defined through their coordinates

Maximum Population at any processor 10	**average**	**best**	**worst**
Genetic algotihms	852,99	675,57	982,83
Complete NGP with 7 processors	550,07	495,66	624,01
Linear NGP with 16 processors	528,52	485,71	601,6
Star NGP with 10 processors	512,18	484,25	545,11
Circular NGP with 13 processors	549,79	521,13	599,56

Networks of Genetic Processors

Solving the Optimization Traveling Salesman Problem (III)

Maximum Population at any processor 20	**average**	**best**	**worst**
Genetic algotihms	676,25	625,8	732,72
Linear NGP with 13 processors	502,35	428,28	553,18
Linear NGP with 16 processors	482,4	453,26	519,58
Linear NGP with 20 processors	503,03	447,66	576,3
Complete NGP with 20 processors	502,46	442,51	567,4
Linear NGP with 20 processors (+ one random generator every 3 processors)	491,07	436,95	541,59

Maximum Population at any processor 30	**average**	**best**	**worst**
Linear NGP with 20 processors	499,71	423,25	539,25
Complete NGP with 20 processors	496,01	457,65	540,99

Networks of Genetic Processors

Work in progress: Applying NGPs to solve multialignment in DNA/protein sequences

```
PLVSS---PLRGEAGVLPFQQEEYEKVKRGIVEQCCHNTCSLYQLENYCN
ALVSG---PQDNELDGMQLQPQEYQKMKRGIVEQCCHSTCSLFQLESYCN
LQVGQVELGGGPGAGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN
PQVGQVELGGGPGAGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN
LQVRDVELAGAPGEGGLQPLALEGALQKRGIVEQCCTSICSLYQLENYCN
PQVAQLELGGGPGADDLQTLALEVAQQKRGIVDQCCTSICSLYQLENYCN
PQVGALELAGGPGAGG-----LEGPPQKRGIVEQCCASVCSLYQLENYCN
PQVEQTELGMGLGAGGLQPLALEMALQKRGIVDQCCTGTCTRHQLQSYCN
  *                   *    *****:***   *   **  ***
```

Networks of Genetic Processors

Work in progress: Applying NGPs to solve multialignment in DNA/protein sequences

To make n-alignment encode the solution as an array of gap/position

String 1	PLVSSLAL-
String 2	P---S-ADG
String 3	P---SL--G

111 100 100 100 111 101 110 110 011

The fitness function consider alignment mismatches and gap penalties