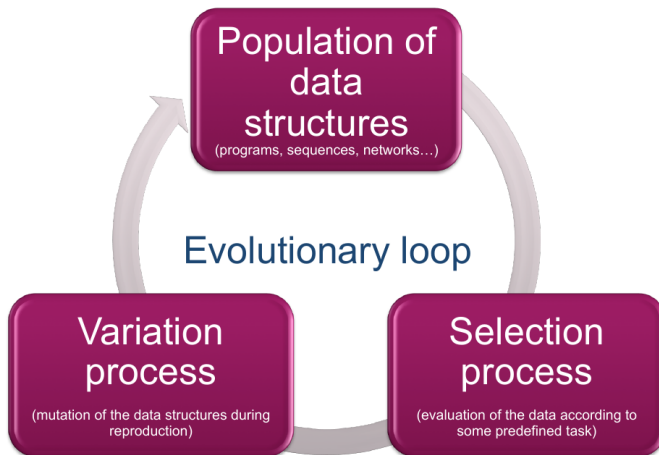


Introducing mini-Aevol: Underlying models, implementation and possible optimizations and parallelization

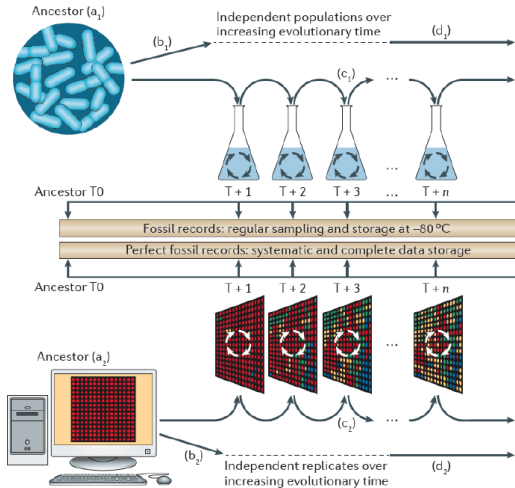
Jonathan Rouzaud-Cornabas

LIRIS / Insa de Lyon – Inria Beagle

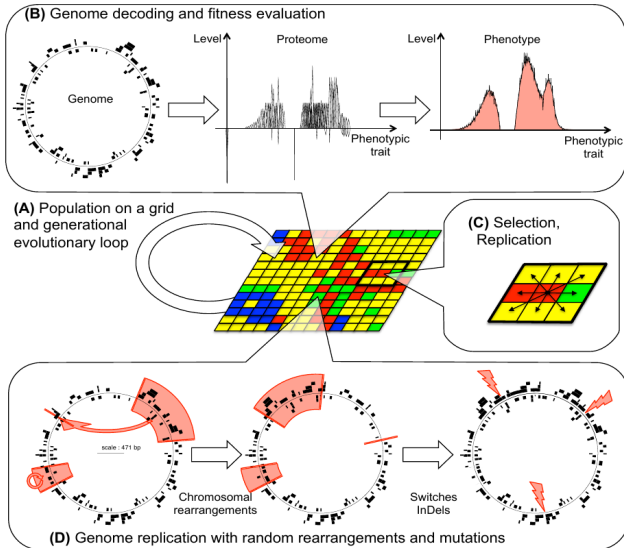
Evolutionary Loop



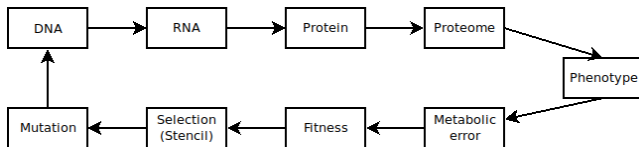
In-silico experimental evolution



The biological model of Aevol

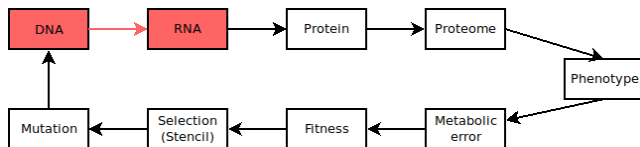


The biological model: A computer science view



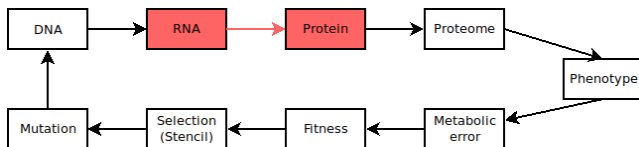
- The workflow is repeat for every cell of the grid
- Each cell contains one and only one organism

The biological model: A computer science view



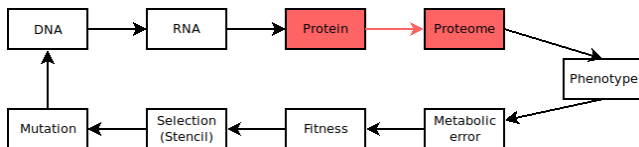
- DNA \rightarrow RNA : **Transcription** : Looking for 2 motifs
 - 1 Promoter (size = 22) : Hamming distance (predefined motif, max. error = 4)
 - 2 Terminator (size = 4) : Exact match
($DNA[pos : pos + 4] == DNA[pos + 10 : pos + 14]$)

The biological model: A computer science view



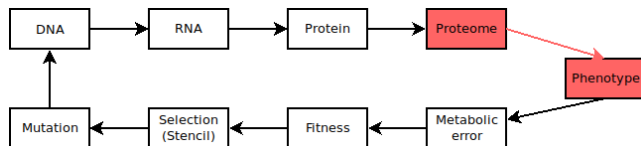
- DNA \rightarrow RNA : **Transcription** : Looking for 2 motifs
- RNA \rightarrow Protein : **Translation** : Looking for 2 motifs
 - 1 Shine Dalgarno (size = 6) + space (size = 4) + Start (size = 3) : Exact match
 - 2 Stop (size = 3) : Exact match

The biological model: A computer science view



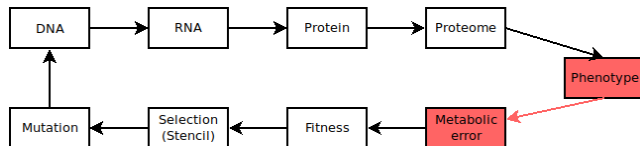
- DNA → RNA : **Transcription** : Looking for 2 motifs
- RNA → Protein : **Translation** : Looking for 2 motifs
- Protein → Proteome : **Folding** :
 - Looking for $N = \frac{\text{Protein_Length}}{3}$ motifs
 - Decoding each triple to addition to M, W, H variables
 - Translation M, W, H into triangle and approx. them into 300 double array

The biological model: A computer science view



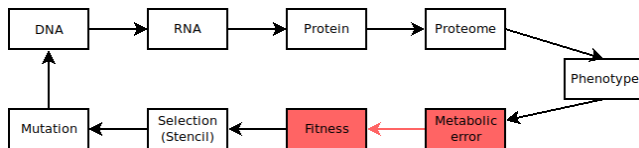
- DNA → RNA : **Transcription** : Looking for 2 motifs
- RNA → Protein : **Translation** : Looking for 2 motifs
- Protein → Proteome : **Folding** :
- Proteome → Phenotype : **Summing arrays** :
 - Looking for $N = \frac{Protein_Length}{3}$ motifs
 - Decoding each triple to addition to M, W, H variables
 - Translation M, W, H into triangle and approx. them into 300 double array
 - Summing into a global 300 double array

The biological model: A computer science view



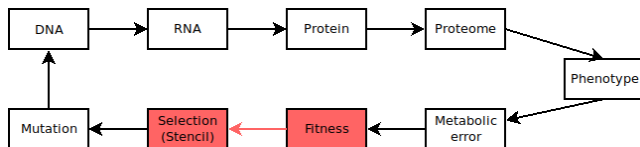
- DNA → RNA : **Transcription** : Looking for 2 motifs
- RNA → Protein : **Translation** : Looking for 2 motifs
- Protein → Proteome : **Folding** :
- Proteome → Phenotype : **Summing arrays** :
- Phenotype → Metabolic error: **Difference between 2 arrays**
 - Phenotypical Target : Pre-defined array built from a sum of gaussians
 - Compute the difference the Phenotype and the phenotypical target

The biological model: A computer science view



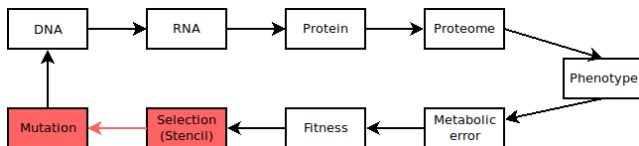
- Metabolic error \rightarrow Fitness:
 - $\exp(-selection_pressure \times metabolic_error)$

The biological model: A computer science view



- Metabolic error \rightarrow Fitness:
- Fitness \rightarrow Selection : **Stencil 9-point 2D**
 - Only kernel requiring data coming from other cells of the grid
 - Fetch fitness of the neighboring cells
 - Use the fitness proportionate algorithms to select the organism that will reproduce in the cell
 - Delete the old organism
 - Copy the DNA of the organism into the cell

The biological model: A computer science view



- Metabolic error → Fitness:
- Fitness → Selection : **Stencil 9-point 2D**
- Selection → Mutation : **Modify the DNA**
 - Switch of the value of up to 6 value in the DNA array

Aevol VS mini-Aevol

- Aevol VS micro-Aevol SLOC : 87,000 VS 2,800
- Simplify model
 - Only a strand of DNA (and not too) *i.e.* the DNA is read one-way and not both-way
 - A reduce mutation sets: no mutation that change the size
 - A lot of advances features are missing: no plasmid, 4bp DNA, ...
- Implementation
 - Lot less robust to input errors
 - Can not change model parameters during an evolution
 - No phylogenetic tree
 - No postprocessing tool

Optimizations

- Working only on diff (where the DNA has changed during the mutation) *i.e.* do not reparse the whole DNA at each generation (available in the git repository for sequential CPU)
- Remove the modulo that allows to simulate circular DNA (copy a part of the beginning of the DNA at the end to do it)
- Change the coding of the DNA from a char array to a bitset
- Vectorization of the comparison between DNA and motifs
- Search multiple motifs at once through vectorization
- Evaluate classic mutation operator (malloc+memcpy+mutation) and backtrace mutation operator (malloc+copy)
- One large 1D structure (merge all DNAs into a big DNA) to ease load balancing
- Include metadata (promoter, terminator, start, stop) into a bitset

Parallelization

- OpenMP (parallel for and/or tasks with or without dependency)
 - CUDA
 - OpenACC
 - MPI
 - Hybrid e.g. OpenMP+OpenACC, OpenMP+MPI, ...
-
- Implement the stencil (*i.e.* asynchronous time between grid cells)

Methodology

- 1 Use GIT (or another versioning software)
- 2 For each optimization/parallelization, tests its performance, scalability (weak and strong) and create visualization of them
- 3 You need to verify the reproducibility of your code *i.e.*, same parameters must lead to same results

Simulation parameters:

- Size of the DNA ($-g$): 500, 5000, 50,000 (you can go up to 500,000)
- Population size ($-w$ and $-h$) : 32x32, 128x128, 256x256 (you can up to 1024x1024)
- Mutation rate ($-m$) : 0.0001, 0.00001, 0.000001