Flibs'NLogo ODD Protocol

Cosimo Leuci

The model description follows the ODD (Overview, Design concepts, Details) protocol for describing individual- and agent-based models (Grimm et al. 2006[1], 2010[2]).

The model is implemented in NETLOGO v. 6.1.1 (Wilensky 1999; freely downloadable from http://ccl.northwestern.edu/netlogo/download.shtml).

1. Purpose

Flibs'NLogo implements in NetLogo modelling environment, a genetic algorithm whose purpose is evolving a perfect predictor from a pool of digital creatures constituted by finite automata[3] or flibs (finite living blobs) that are the agents of the model. The project is based on the structure described by Alexander K. Dewdney in "Exploring the field of genetic algorithms in a primordial computer sea full of flibs" from the vintage Scientific American column "Computer Recreations"[4]

As Dewdney summarized: "Flibs […] attempt to predict changes in their environment. In the primordial computer soup, during each generation, the best predictor crosses chromosomes with a randomly selected flib. Increasingly accurate predictors evolve until a perfect one emerges. A flib […] has a finite number of states, and for each signal it receives (a 0 or a 1) it sends a signal and enters a new state. The signal sent by a flib during each cycle of operation is its prediction of the next signal to be received from the environment"[5]

[1] V. Grimm, U. Berger, F. Bastiansen, S. Eliassen, V. Ginot, J. Giske, J. Goss-Custard, T. Grand, S. K. Heinz, G. Huse, A. Huth, J. U. Jepsen, C. Jørgensen, W. M. Mooij, B. Müller, G. Pe'er, C. Piou, S.F. Railsback, A. M. Robbins, M. M. Robbins, E. Rossmanith, N. Rüger, E. Strand, S. Souissi, R.A. Stillman, R. Vabø, U. Visser, D. L. DeAngelis *A standard protocol for describing individual-based and agent-based models* Ecol. Modell. (2006) 198, 115–126. (doi:10.1016/j.ecolmodel.2006.04.023)

[2] V.Grimm, U. Berger, D. L. DeAngelis, J. G. Polhill, J. Giskee, S. F. Railsback *The ODD protocol: A review and first update* Ecological Modelling 221 (2010) 2760–2768 (doi:10.1016/j.ecolmodel.2010.08.019)

[3] Brian Hayes *On the finite-state machine, a minimal model of mousetraps, ribosomes and the human soul* Scientific American, 249, No. 6 (1983), pages 19-28. Available at: https://tinyurl.com/ycukjtor

[4] A. K. Dewdney *Exploring the field of genetic algorithms in a primordial computer see full of flibs* Scientific American 253, No. 5 (1985), pages 21-33. Partially available at: https://tinyurl.com/y84bnear

[5] A. K. Dewdney *The king (a chess program) is dead, long live the king (a chess machine)* Scientific American 254, No. 2 (1986), p. 21. Available at: https://tinyurl.com/ycphyqqm

2. Entities, state variables, and scales

Flibs (Finite living blobs) are the agents of the model and they could be regarded as very simple living beings, potentially able to predict the periodic variations in the simplest environment where they live.

At the hearth of Flibs there is a finite-state automaton whose behaviour can be represented in a state-transition table providing for each flib's state:

1. a binary output of the flibs as a response to an environmental binary input;
2. the consequent new flib's state.

In the following table you can see an example of a three-state flib:

STATE	INPUT 0		1	
A	1	B	1	C
B	0	C	0	B
C	1	A	0	A

So, if a flib is in a C state and get an environmental signal equal to 0, according to the table above it responds generating a 1 and entering into the state A. Every flib has got its own state-transition table, but it is represented as a linear string called "chromosome"; the corresponding string of the table above would be:

1B1C0C0B1A0A

Replacing the letters of states with numbers (A = 0, B = 1, C = 2 ... *etc.*) we obtain definitively the codified chromosome as appears in the program:

111202011000

Every character of the string can be regarded as an "allele" and we will utilise the name "locus" for its position along the chromosome (the first locus has index 0)

As a consequence of chromosome architecture, the alleles number for each chromosome string will be four times greater than the number of flib states. If the flib is in the state with numerical value s and the environmental input is i, then its future state will be easily found in the table; the locus of this element is given by:

$$4s + 2i + 1.$$

Similarly, the value of the output variable generated by the flib is stored in the locus:

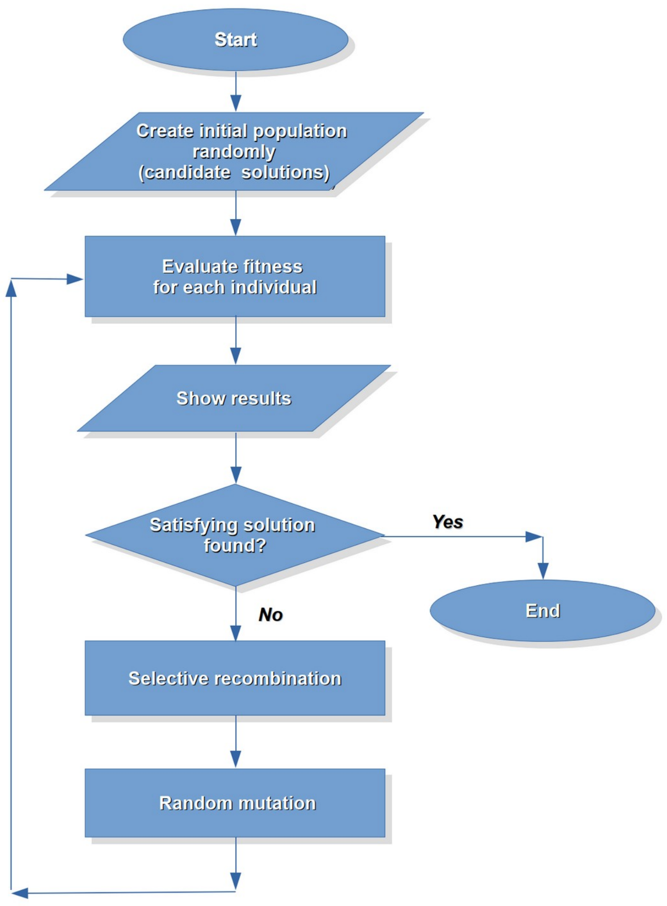
$$4s + 2i.$$

The chromosome and the initial state are important parameters defining a flib. Another important parameter is the fitness: it is computed in along a generation lasting 100 iterations and represents the percentage of correct predictions. A flib give a correct prediction when its output signal coincides with the next environmental input signal.

As an input signal for flibs, the algorithm uses a given sequence of binary digits. This sequence can be defined by the user and it is cyclically read during a generation; at the beginning of a new generation the environmental sequence as the flib's state come back to their starting position.

3. Process overview and scheduling

The process follows the structure of a genetic algorithm as shown below:


Start
Create initial population randomly (candidate solutions)
Evaluate fitness for each individual
Show results
Satisfying solution found?
Yes
End
No
Selective recombination
Random mutation

In more detail, for Flibs'NLogo we have:

1. A population of random flibs is generated. The flibs number and the number of their internal states can be previously set by the user.
2. In a new generation consisting of one hundred iterations, the number of changes in the environmental state predicted correctly by each of these flibs is calculated.
3. The measure of the flibs performance is assigned to the variable named fitness.
4. The worst and best predictors in the population are determined on the basis of individuals' fitness
5. The two values and the average of population fitnesses, as well as fitnesses distribution are plotted.
6. If one of the flibs reaches a 100% prediction level the algorithm stops (however the program can be terminated by the user in every time).
7. The chromosome of a randomly chosen flib is crossed with the one of the best predictor, through a recombination operator. Consequently, the chosen flib is replaced by a hybrid.
8. A random point mutation is applied to a flib according to the frequency set by the user.

9. Some detail about genetic variations occurred at the end of the current generation are displayed.
10. Go to the item (2).

4. Design concepts

Genetic algorithms design assumes that optimization of a feature can be a consequence of two processes: casual mutation and selective genetic shuffling (usually by homologous recombination) as occurs in breeders work; as previously seen, to select (one of) the best individuals, the algorithm must compute the fitness function for each of them. Generation after generation, the optimized trait emerges by Darwinian selection: the trait that Flibs'NLogo tries and develops in at least one flib, is the ability to predict the stable sequence of environmental binary inputs.

We could regard all the process as an attempt to adapt to the given environment as well as an elementary form of cognition process; indeed, some scholars (e.g. Karl Popper[6]) consider scientific thought as an extension of organic evolution. In the perspective of autopoietic view of life[7], we can say that flibs approach the structural coupling with its environment by the genetic algorithm.

5. Initialization

Flibs are created randomly but coherently with the general chromosomes architecture and the states' number set by the user, according to the following algorithm:

➔ an iteration randomly assigns to every element of the chromosome string:
  - one of the two environmental states, if the locus of an element is even;
  - one of the possible flib states, if the locus of an element is odd.

6. Input data

Not only flibs states but also flibs number, also recombination rate and mutation rate can be set by sliders in the user interface of Flibs'NLogo. Furthermore, a text box allows to input the binary string (just 0 and 1 digits are accepted) of the environmental sequence of states.

We can observe that the sequence length is inversely proportional to the difficulty that the optimization process faces in tuning the flibs' behaviour; as Dewdney warns, a flib will never be able to predict an environmental sequence longer than the double of its number of internal states, then no flib will be able to predict a random environmental string, that is a binary sequence having an infinite period[8].

[6] Karl Popper, *All Life is Problem Solving* (1999)

[7] Humberto R. Maturana, Francisco G. Varela, *De Maquinas y Seres Vivos* (1994)

[8] A. K. Dewdney 1986, *op. cit.,* p. 21.

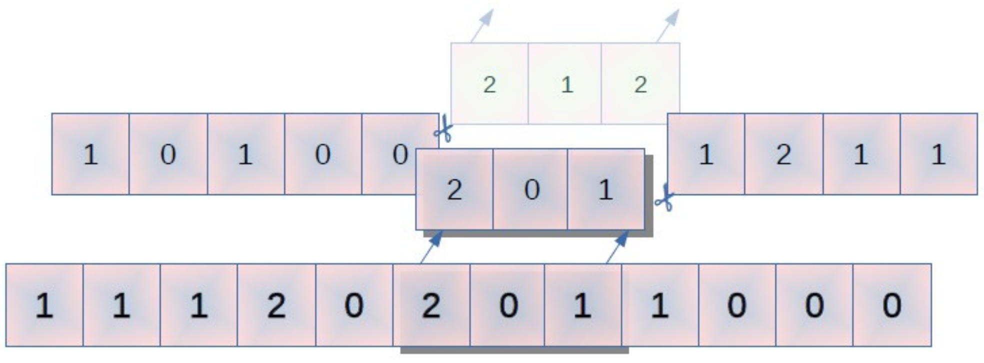
7. Submodels

Flibs'NLogo has got three operators corresponding to three submodels:

1. evaluation of fitness function
2. recombination
3. mutation

1. EVALUATION OF FITNESS FUNCTION. Resuming what we said before, the best flib is the one with a maximum number of correct predictions of the binary signals coming from the environment. When a new generation is ready, all the flibs state are reset to zero; then the fitness of flibs is determined by testing their prediction capability along 100 cycles and the number of correctly predicted inputs is computed.

2. RECOMBINATION. In spite of the model described by Dewdney, Flibs'NLogo exhibits a different kind of homologous recombination, similar to bacterial recombination related to conjunction events, more than the eukaryotic meiotic crossing-over: indeed just the receiving chromosome varies its genetic kit; on the contrary, the selected best flib replicate part of its chromosome (the donor chromosome) into the corresponding loci of the second one (the recipient chromosome) so that no flib is removed from the soup, but one of them changes part of its chromosomal sequence by hybridization (see figure below); the two split points are chosen randomly. The mechanism was already used in the model "Minimal Genetic Algorithm"[9].



3. MUTATION. Recombination events occur in random chromosomes loci of partly random chosen flibs, mutation events occur in totally random flibs and loci; the mutations are always point-mutation: according to Dewdney suggestions, a locus of a certain chromosome is chosen randomly; if its position is even (so, its content is the binary value generated by the flib, as response to environmental stimulus) the value of the content is flipped; if the locus position is odd (so, it contains the next flib state) the current state of the flib is replaced by the next one, i.e. it is increased modularly of one unit.

[9] Cosimo Leuci, (2019, September 03). "Minimal Genetic Algorithm" (Version 1.0.0). CoMSES Computational Model Library. https://doi.org/10.25937/7kta-pf44