Natural Computing

Lecture 6

Michael Herrmann
mherrman@inf.ed.ac.uk
phone: 0131 6 517177
Informatics Forum 1.42

07/10/2011

Hybrid GA and Practical Applications

Lamarckism

Characterised by

- Inheritance of acquired traits
- Use and disuse determine characteristics

More specifically, Lamarck provided a systematic theoretical framework for understanding evolution as the interplay of two processes

- **A complexifying force:** in which the natural, alchemical movements of fluids would etch out organs from tissues, leading to ever more complex construction regardless of the organ's use or disuse. This would drive organisms from simple to complex forms.
- **An adaptive force:** in which the use and disuse of characters led organisms to become more adapted to their environment. This would take organisms sideways off the path from simple to complex, specialising them for their environment.

wikipedia on Jean-Baptiste Lamarck

The Baldwin effect

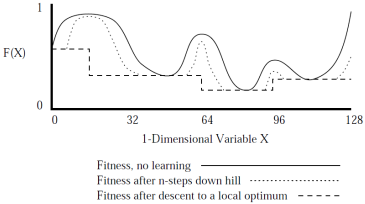
- "A new factor in evolution" (James Baldwin, 1896)
- Selection for learning ability (rather than relying only on fixed abilities from the genes)
- Increased flexibility: Robustness to changes in the environment (i.e. changes of the fitness function)

[University of Toronto]

- Selective pressure may lead to a translation of learned abilities into genetic information!
  - Learning has a cost
  - If learning of the same tasks increases fitness over many generations then those individuals have a relatively higher fitness that produce (parts of) these results by their genetically fixed abilities

Computational studies

- Hinton & Nowlan: How learning can guide evolution (1987) (see M. Mitchell, Chapter 3)
  - Binary genome plus undecided bits which are set in "life" by learning
- Whitley, Gordon & Mathias: Lamarckian evolution, the Baldwin effect and function optimisation (1994)
- Standard GA (elitist) plus: Lamarckian evolution (editing strings) or Baldwinian evolution (adaptation process before fitness determination)



[technically in both cases: hill-climbing in the fitness landscape]

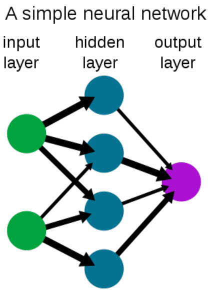
- Lamarckian faster, but Baldwinian less likely to be trapped by local optima

Hybrid GA: Evolving Neural Networks

- Reminder of neural networks
  - Inspired by the function of neurons in the brain
  - Universal function approximators
  - Used for: Prediction, classification, pattern completion, control, modelling the brain
- In connection to GA
  - Evolving weights
  - Evolving network topology
  - Evolving grammars

Neural networks: Basics

- Nodes (neurons) and connections (synapses)
- Weights (efficacies) attached to the connections
- Output from a node (spike or firing rate) depends on the weighted sum of inputs to the node
- Non-linear activation (threshold) function
- Often arranged in layers (areas) or as a recurrent structure
- Training algorithms to determine the weights (error-backpropagation)



[wikipedia]

A simple feed-forward neural network

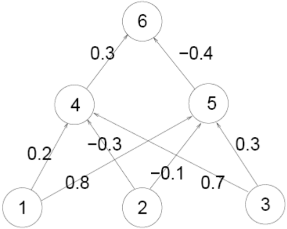
- Activity of a node y_i depends on inputs x_j
- Weight w_{ij} from node j to node i
- Each node (apart from the input nodes) takes the weighted sum of its inputs and feeds the result through a sigmoid function, $y_i = \frac{1}{1+e^{-u_i}}$ where $u_i = \sum_{j=1}^{L} w_{ij} x_j$
- Output activity is often input to another node ($x_k = y_i$)
- Given a number of examples $\left(x_{\text{input}}^{(m)}, y_{\text{output}}^{(m)}\right)$, $m = 1, \ldots, M$ the networks should generate outputs $y\left(x_{\text{input}}^{(m)}\right)$ which are similar to $y_{\text{output}}^{(m)}$, i.e. minimize (typically by gradient decent: error backpropagtion)

$$\frac{1}{M} \sum_{i=1}^{M} \left(y\left(x_{\text{input}}^{(m)}\right) - y_{\text{output}}^{(m)} \right)^2$$

Evolving weights

- Evolve weights rather than train the network directly
- Alternative to error backpropagtion
- Classification of underwater sonic recordings (Montana & Davis, 1989)
  - network topology:
    - 4 input nodes
    - 7 nodes in first hidden layer
    - 10 nodes in second hidden layer
    - 1 output node
    - fully connected
  - 18 extra thresholding connections (biases)
  - total weights 126 → GA chromosome as a list 126 real valued weights

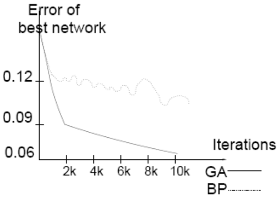
Representation of a neural network



- **Chromosome:** (0.3, -0.4, 0.2, -0.3, 0.7, 0.8, -0.1, -0.3)
- **Building blocks:** all incoming weights to a given unit
- **Mutation:** for **each** incoming link to a chosen node, add a (different) random value between -1 and +1
- **Crossover:** for each non-input node, cross **all** the weights of parent 1 with **all** the weights of parent 2 (Montana-Davis crossover)

Results of the weight evolution

- Reward function: Match of desired output (from examples) and actual network output
- GA were better than (supervised) BP on some tasks



- Selection as 'unsupervised' learning (individual networks are not changed here in order to produce a better output)
- Useful if only sparse reinforcement is available (e.g. if the networks controls a robot in an unfamiliar environment)
- Backpropagation does not work well if subsequent inputs are correlated and may suffer from local minima, so GA may have an advantage in some cases

Evolving topology (subject to constraints)

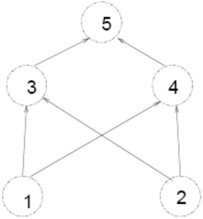
- Choosing a network topology is hard
- Leave weight adaptation to the neural network learning and determine the connectivity by a GA

Connectivity matrix (cf. Miller, Todd & He

from unit

to unit

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	1	1	0	0	0
4	1	1	0	0	0
5	0	0	1	1	0



0 if not connected
1 if connected = learnable

Chromosome: 00000 00000 11000 11000 00110

Mutation: bit flipping

Crossover: exchange whole rows

Constraints: Feed-forward topology (lower-triangular matrix), no self-connections (zero diagonal)

Grammatical Encoding

Grammatical encoding of the linkage matrix (here for XOR)

$(S \rightarrow A\,B\,C\,D \,|\, A \rightarrow c\,p\,a\,c \,|\, B \rightarrow a\,a\,a\,e \,|\, \ldots)$

$$S \rightarrow \begin{array}{cc} A & B \\ C & D \end{array} \rightarrow \begin{array}{cccc} c & p & a & a \\ a & c & a & e \\ a & a & a & a \\ a & a & a & b \end{array} \rightarrow \begin{array}{cccccccc} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

Generate connection matrix from grammar. If at the end of rewriting there are still non-terminal nodes, that node is "dead" (not connected)

Grammatical Encoding

Develop chromosome (genotype) into network (phenotype) and train for fixed number of training episodes

Fitness: Error at the end of training

Problems in direct encoding

Fixed connections: as size of matrix grows, chromosome size grows

Many generation until convergence

Cannot encode repeated patterns (weight sharing), especially with internal structure

Advantages of grammars

Representation of large connection matrices in compact form

Shorter encoding → faster search

Variable topologies including recurrent connections

Empirically better than direct encoding (e.g. auto-encoder problem)

Neuro-Evolution though Augmenting Topologies (NEAT)

Evolve by changing the connection weights, turning links on and off and also by adding links and nodes (in the middle of an existing link) (in the mutation stage)

Start off simple, become more complex (with a punishment for complexity in the fitness function)

Crossover: Match up parts of the network coding or similar traits

Competing conventions: Permuting a hidden nodes of a network does not change the output or the function computed by the network

So: give each gene an innovation number: the next unused integer with a new structure is added. So the algorithm can match up parts of networks inheriting this gene in future generation

Neuro-Evolution though Augmenting Topologies (NEAT)

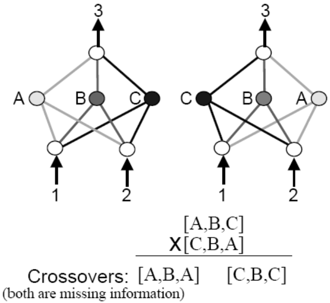
Inherit matching genes from each parent with equal probability.
Inherit non-matching genes from fittest parent

Can also split into species (“islands”) based on difference between chromosomes (based on number of matching genes and other metrics (e.g. output errors). Preserves new topology for a while so that it has a chance to optimise its structure only in competition with similar members

Works well on pole-balancing. Also applied to game of Go

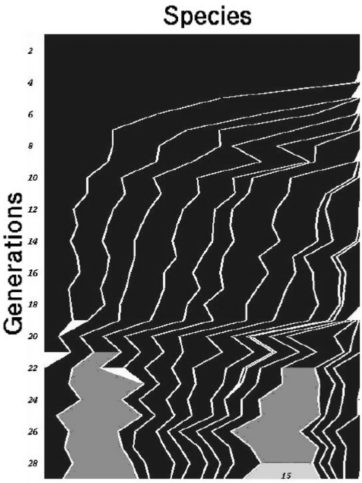
Stanley & Miikulainen Evolutionary Computing 10, 99-127 (2002)

Neuro-Evolution though Augmenting Topologies (NEAT)



Above: Justification of "innovation numbers"

Right: Visualisation of speciation in a double pole balancing task

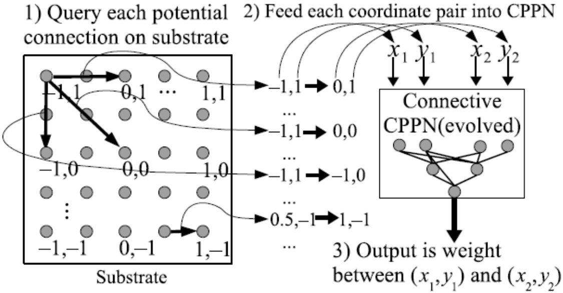


Stanley & Miikulainen Evolutionary Computing 10, 99-127 (2002)

Hyper-NEAT

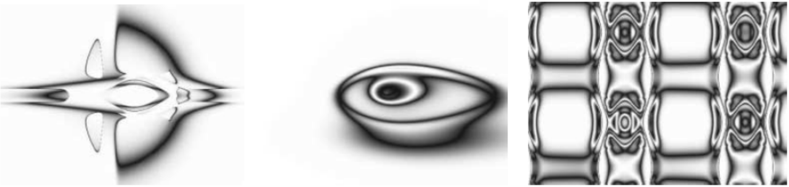
Hypercube-Based Encoding for Evolving Large-Scale Neural Networks

- Compositional pattern-producing networks: Patterns with symmetries and repeating motifs (canonical microcircuit)
- Scale ANNs to new numbers of inputs and outputs without further evolution.
- Recognition of symmetries and repetitions in 2D images by a hierarchical encoding scheme

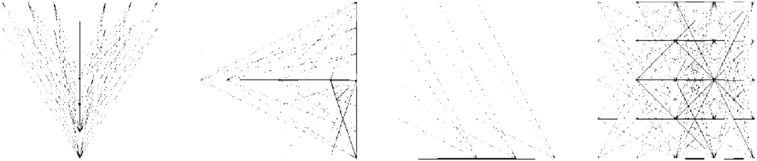


Stanley, D'Ambrosio, Gauci (2009)

Hyper-NEAT



(a) Symmetry (b) Imperfect Symmetry (c) Repetition with Variation

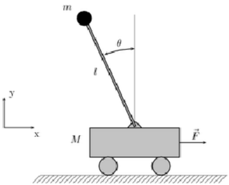


(a) Symmetry (b) Imperfect Sym. (c) Repetition (d) Rep. with Var.

Grammatical Encoding (Summary)

- Representation: Represent together what belongs together
- Grammatical encoding allows for re-using building blocks
  - weight-sharing
  - compositionality of behaviour
  - hierarchical representation
- Special operators that respect the rules
- Special protection for important rules
- Nontrivial genotype - phenotype relation

Evolving Neuromodules for Control



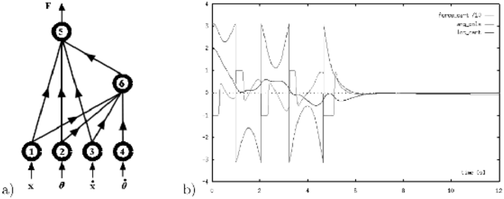


Figure 1: a.) A minimal 4t-class solution w^1 and b.) its effective control: $x(t)$, $\theta(t)$, and $F(t)$ starting from $x_0 = 2.0$ and $\theta_0 = \pi$.

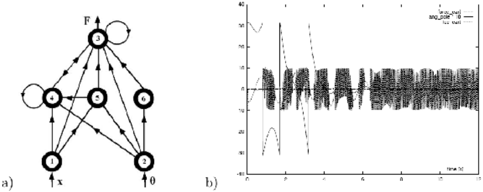


Figure 5: a.) The 2t-controller w^3 solving the swinging-up problem, and b.) cart position and pole angle under its action, starting from $x_0 = 0$, $\theta_0 = \pi$.

F. Pasemann et al. (1999) Evolving structure and function of neurocontrollers. MIS preprint

Evolving Robots Learn To Lie To Each Other

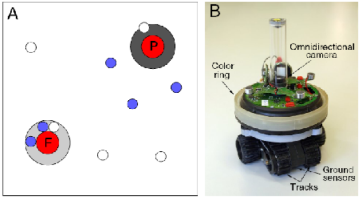


Fig. 1. Experimental setup. (*A*) A food and poison source, both emitting red light, are placed 1 m from one of two opposite corners of the square (3- × 3-m) arena. Robots (small circles) can distinguish the two by sensing the color of the circles of paper placed under each source by using their floor sensors when driving over the paper. (*B*) The robot used for the experiments is equipped with two tracks to drive, an omnidirectional (360°) vision camera, a ring of lights used to emit blue light, and floor sensors to distinguish food and poison sources (see ref. 14 for details).

- 1,000 robots divided into 10 groups
- Each robot had a sensor, a blue light, and a 264-bit binary genome encoding a controller
- Initial population: turn the light on at food resource, supporting also other robots
- Positive fitness points for finding and sitting at the good resource, negative for being near the poison
- 200 fittest robots are selected, recombined and mutated
- Near optimal fitness after 9 generations
- A limited amount of food results in overcrowding

After 500 generations: 60 % of the robots kept their light off near food
Other robots adapted to this and developed an aversion to the light

Sara Mitria, Dario Floreano and Laurent Keller (2009) The evolution of information suppression in communicating robots with conflicting interests. PNAS