Selected Applications of Sub-Symbolic AI Methods

Keith L. Downing

The Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
keithd@idi.ntnu.no

March 27, 2011
The first EA, now known as an evolutionary strategy, was a mechanical device, not a computer algorithm.

Goal: Minimize drag on a 6-plate, 5-hinge surface.

Solution: All angles = 0 (similar to OneMax problem)

1+1-ES with wind-tunnel for fitness (drag) measurement.

# Possible Configurations = $51^5 = 345,025,251$

The experimentum crucis - Drag minimization of the kink plate
- 1+1-ES over 200 generations.
- 45 physical models were built and tested during the process.
- Schwefel was the first to program an ES, but fitness testing still needed to be done physically; no good simulators.
Task: Sort a 16-element list using minimal comparisons

- Sorting algorithms are characterized by compare-and-swap operations.
- E.g. (5, 8) means compare $X(5)$ to $X(8)$ and swap if $X(5) > X(8)$
- The trick = do comparisons in the right sequence.

History of 16-sort progress

- 65 comparisons (1962)
- 63 comparisons (1964)
- 62 comparisons (1969)
- 60 comparisons, optimal, (1969)

Bubblesort = $15 + 14 + 13 + \ldots + 1 = 120$ comparisons
One pair of chromosomes:

Diploid Genome: 15 pairs of chromosomes, each of which encodes 4-8 comparisons.

So the genome encodes between 60 (4 x 15) and 120 (8 x 15) comparisons.

Diploid Crossover (just like in biology): parent chromosomes do not cross until gametes are produced.

Fitness Test = several lists of unsorted integers.

Fitness = number of cases sorted correctly.
Best result without coevolution = 65 comparisons.

Why do smaller comparison sets evolve when size is not part of the fitness function? Answer: Good comparisons become popular, get copied a lot, and end up on corresponding chromosome pairs.

Adding **Coevolution** ⇒ add population of unsorted lists that compete against the sorting schemes.

Fitness(unsorted list) = number of sorting schemes that fail to correctly sort it.

Result using coevolution = 61 comparisons!

First convincing argument for coevolution in evolutionary computation.
The Growing Circuit

Keith L. Downing | Selected Applications of Sub-Symbolic AI Methods
Nearly Finished

Keith L. Downing
Selected Applications of Sub-Symbolic AI Methods
Human Competitive Results

1. Re-invention of many 20th-century electronic circuits.
3. Automated synthesis of PID controllers.
5. Design of metabolic pathway models.
6. Design of polymer optical fibers.
8. Patented antennas.

* The first 5 involve Koza’s 1000+ node Beuwolf cluster.
**Given:** A serial program.

**Find:** A parallel version that does not violate code dependencies.

**Method:**

- GP population = Transformation Trees
- Apply tree to serial code to produce parallel code.
- Assess fitness in terms of program speed (pos) and violated dependencies (neg).

### Code Dependency

1. \( A = B + C \)
2. \( D = A + E \) (1 and 2 can’t be run in parallel, since 1 modifies A while 2 uses A.)
3. \( X = W + Y \)
4. \( Z = W + V \) (3 and 4 can be parallel despite sharing W, since neither modifies it.)
GP Tree Primitives

- **S(x):** Break sequence into 2 chunks, which are still run serially (so far) but can be treated independently during later parallelization steps. $x = \%$ in first chunk.

- **P(x):** Break sequence into 2 parallel chunks. $x = \%$ in first chunk.

- **Fseq, Lseq:** Pick out first (F) or last (L) item and treat it as a separate serial chunk from the rest.

- **Fpar, Lpar:** Pick out first (F) or last (L) item and run it in parallel with the rest of the items.

- **Shift:** Delay the starting time for a chunk by one timestep.

- **Null:** Do nothing more with the chunk.

- **ParNull:** Parallelize each item in the chunk, and then do nothing more with the items.

- **Loop Transformations:** A whole set of operations for optimizing loops. These are housed in a separate linear genome.
An Evolved Parallelization Scheme

*Each letter = a line of code*

```
(ABCDEFGH) (ABCD) (EFGH) (A)(BCD) (EFGH) A(BCD) (EFGH) (FGH)
A  B  C  D  E  FGH
```
Steven Rooke’s Evolutionary Art

Uses GP to map \((X, Y)\) to colors. (See http://sroke.com/)
Weighted Function Graphs

Keith L. Downing

Selected Applications of Sub-Symbolic AI Methods
Compositional Pattern-Producing Network (CPPN)

K. Stanley (2006, 2007) - CPPNs
J. Secretan, K. Stanley, et. al. - PicBreeder (picbreeder.org)

**Applications:** Checkers, Othello, Robot Control.
Neural Networks in Control Systems

The neural network can serve as:

- the controller component, performing $e \mapsto u$ mapping.
- a model of the physical system (a.k.a plant): does $u \mapsto y$ map.
- MANY combinations are possible.

ANN learns to do feedforward control: it maps $y^* \mapsto u$.

The recommendation (for $u$) from the feedback controller serves as an error signal to the ANN's learning algorithm (e.g. backprop).

After training, the ANN can replace the feedback controller.
- ANN used to control the plant
- ANN used to model the plant
- Error = difference between actual and predicted plant output
- Both ANNs can be trained off-line, using data from the normal operation of the plant.
A Neuro-Evolutionary Approach to MAV Control, Salichon & Turner (2010)

- Tiny (insect-to-bird size) airplanes and helicopters are useful for exploring small or difficult-to-access areas.
- But they are hard to control due to strong perturbation by wind gusts and turbulence.
- Evolved 3 separate controllers for altitude, heading and roll.
- 3-5 times better performance than standard PID controllers, and with much less tuning effort.
- No model of the MAV or world was needed (as in other control methods). Hence, it’s easy to re-evolve and re-deploy in new areas.
1 Train the ANN (typically using backprop) on historical data to learn 
\[ X(t_{-k}), X(t_{-k+1}), \ldots, X(t_0) \mapsto [X(t_1), \ldots, X(t_{m-1}), X(t_m)] \]
2 Use to predict future value(s) based on the past \( k \) values.

Sample applications (Ungar, in *Handbook of Brain Theory and NNs, 2003*)
- Car sales
- Airline passengers
- Currency exchange rates
- Electrical loads on regional power systems.
- Flour prices
- Stock prices (*Warning*: often tried, but few good, documented results).
Brain-Computer Interfaces (BCI)

1. Ask subject to think about an activity (e.g., moving joystick left)
2. Register brain activity (EEG waves - non-invasive) or (Neural ensembles - invasive)
3. ANN training case = (brain readings, joystick motion)

Sample applications (Millan, in *Handbook of Brain Theory and NNs*, 2003)

- Keyboards (3 keystrokes per minute)
- Artificial (prosthetic) hands
- Wheelchairs
- Computer games
A New Modular GP for Finding Attractive Technical Patterns in Stock Markets, (Lee & Moon, 2010)

**Goal:** Find boolean patterns (expressed by GP) that classify good stocks to BUY at particular times.

**GP Primitives**
- $p_o(t) = \text{opening price on day } t, \text{ where current day } = 0.$
- $p_c(t) = \text{closing price on day } t.$
- $p_h(t) = \text{highest price on day } t.$
- $v(t) = \text{trading volume on day } t.$
- $MA_n(t) = \text{moving n-day average ending at day } t.$

**Sample Technical Patterns**

\[ 1.1 \times p_o(-1) \leq p_o(0) \land MA_{40}(0) \geq MA_{20}(-1) \]

\[ v(0) \geq 2 \times v(-7) \lor p_h(0) \geq 1.5 \times p_h(-7) \]
Fitness(Technical Pattern) $\propto$ Profitability

**Expected Earning Rate of pattern r**

$$E_k(r) = \frac{1}{|R(r)|} \sum_{(i,j) \in R(r)} \frac{p_c(i,j + k)}{p_c(i,j)}$$

where:

- $R(r) = \{ (i,j) : r \text{ matches company } i \text{ on day } j \}$
- $p_c(i,j) = \text{closing price of company } i \text{ on day } j$

**Fitness**

If $|r| < M$ (i.e. $r$ doesn’t have too many terms) and $|R(r)| \geq m$ ($r$ has many matches to company-days), then:

$$f(r) = \frac{1}{n} \sum_{k=1}^{n} E_k(r)$$

Otherwise, $f(r) = 0$.

This averages the expected earning rate over all window sizes from 1 to $n$ (the smoothing factor).
Modular GP

GP Trees composed of pre-defined modules, such as:

\[ mp_0(t) > p_c(t) \land np_0(t) < p_c(t) \]  where \( m, n \in (0.99, 1.0) \) and \( t \in [-4, 0] \)

50+ modules for this domain

Results

- Best patterns are small (\( \leq 5 \) modules).
- GP found patterns with \( \geq 8\% \) profit on all test sets.
EAs in large-scale open pit mine scheduling (Myburgh & Deb, 2010)

- 3d models with $O(10^{5-6})$ blocks containing ore and/or waste.
- **Goal:** Bring up the most ore and the least waste as efficiently as possible while satisfying constraints induced by levels and toe blocks.
- A complete level need not be removed before another, but a) all blocks above block A must be removed before A, and b) a certain *mining width* of blocks must be removed on a level to allow equipment to operate there (and thus dig below that level).
- Conventional optimization techniques can’t handle the size of this problem: too many blocks.

The Pit Mine

Unmined

Mined

Toe Block

Ore

Waste

Toe Block = 1st block that must be removed from its level.
Evolving Mining Schedules

Methods

- Specialized crossover operators needed to avoid block redundancies and omissions in the children.
- Period = a sequence of blocks needed to achieve a given total-ore target. They can vary in length.
- Fitness = inverse function of total waste mined during the periods.

Results

- Good results obtained only when chromosomes are incrementally evolved, one period at a time: once good values are found for period 1, move on to period 2, etc.
- Many real-world, Australian, mine schedules determined, of various sizes: 320K blocks, 460K blocks, 900K blocks, etc.
Evolving Viral Marketing Strategies

Stonedahl, Rand & Wilensky (2010)

**Task**

**Given:**
- A network of potential consumers, each of which is connected to some *neighbors*, who can influence his/her purchasing decision. In general, the more neighbors who adopt the product, the better chance that you will also.
- A limited budget for *free samples* of the product.

**Find:**
- a set of consumers to receive a free sample, such that:
- the rest of the consumers will adopt (i.e. buy) the product as quickly as possible.
Adoption Function: $f(t) = \text{probability of adopting}$

$$f(t) = p + q \times \frac{n_a(t)}{n}$$

where:
- $p = \text{external influence}$, $q = \text{neighborly influence}$
- $n_a(t) = \# \text{neighbors that have adopted by time } t$
- $n = \# \text{neighbors}$
Evaluating Seed Potential of Nodes

5 Key Node Properties

- **Degree** - of connectivity to other nodes.
- 2-step - # nodes reachable via 1 or 2 connections
- Avg path length - to all other nodes in the network
- Clustering coefficient - 1 minus fraction of neighbors that are interconnected. High clustering is not conducive to spreading adoption.
- A randomizing factor.

A weighted combination of these factors determines a node’s evaluation.

Nodes are sorted by their evaluations.

The first k nodes in this list become seeds.
Mixed Strategy Encoding

- Strategy = Evaluation function for nodes, which determines seeding.
- Chromosome encodes 2 strategies, the first of which is applied to any given node with prob p (also evolved).
Evolving and Testing Seeding Strategies

Fitness Testing

- Use evolved seeding strategy to seed the network.
- Test each node for adoption. Update status if it adopts.
- Keep updating nodes until all have adopted.

Fitness = Net Present Value (NPV) of the Adoption Network

\[ NPV(G, S, f(t)) = \sum_{t=0}^{\infty} a(t)\lambda^t \]

- \( G \) = the social network (graph)
- \( S \) = fraction of nodes to seed (evolved)
- \( f(t) \) = the adoption function - there are several options.
- \( a(t) \) = # that adopt AT time \( t \)
- \( \lambda \) = discount rate (e.g. \( \lambda = 0.9 \))

So the longer that it takes the whole network to adopt, the lower the fitness.
Several Network Types

- Random
- Lattice
- Small World
- Preferential Attachment
- Twitter

They had hoped to find interesting relationships between network types and the seeding strategies for each. However, they found very little of significance. Still, the application is quite unique, and interesting. The general approach could transfer to other domains.