Tree Structured Genetic Programming

First demonstrated as a practical working system by John Koza [1, 2]

Generic Objective

- Evolve model given suitable data
  - Supervised ($\mathbf{x}^p$, d^p)
  - Reinforcement ($\mathbf{x}^p$, r^p)
- Requires *a priori* specification of
  - Suitable language
  - Cost Function (goal)
  - Learning Parameters

Basic Preparatory Steps

1	Select Terminal Set (Arg 0)
2	Select Functional Set (Arg > 0)
3	Define fitness function
4	Set parameters (population, search and selection operators)
5	Establish stopping criterion

Fitness Function – Error Criteria (representation independent)

- Consider a supervised learning context
  - Implies evaluation over a training set
  - Training set must be representative of the overall problem
  - Fitness calculated for individual i, at generation t.
- Distance Based Criteria
  - Raw fitness
  - Standardized fitness
  - Normalized fitness

Raw Fitness $\langle r_{t,i} \rangle$

- Typically problem specific
  - SSE = SUM(e^2)
  - MSE = SUM(e^2) / P
  - RMS = √ (SUM(e^2) / P)
    - e is the Euclidian distance d – y, d is the desired pattern and y is the response evaluated from GP individual i.
    - P is the number of patterns in the training set.
  - Count of exact matches
    - Often referred to as the number of "Hits".
    - Might need a Wrapper to interpret GP output.

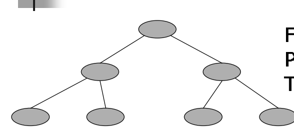
Standardized Fitness $\langle s_{t,i} \rangle$

- Fittest individual has a fitness of zero
- Worst individual has a fitness >> 0 or << 0(not both in same problem!)
- No longer problem specific
  - E.g. count based fitness needs re-expressing to satisfy this criteria.

Normalized fitness $\langle n_{t,i} \rangle$

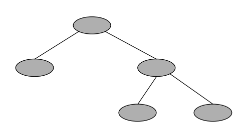
- Limited to the range 0 – 1
- Larger for fitter individuals
- Sum of normalized values is UNITY.
  - $n_{t,i} = (a_{t,i}) / [\mathrm{SUM}(a_{t,k})]$

Initialization – Tree Structured



Full – always Produces a balanced Tree

Grow – typically produces a tree with unequal branch depths.



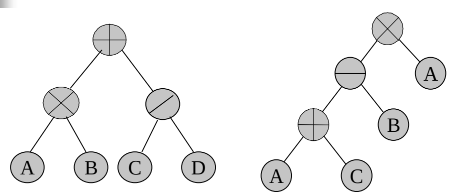
Initialization – Tree Structured

- Ramped half-half method
  - Specify maximum depth, say 6
    - 20% have max depth of TWO, of which 50% are created using GROW and 50% using FULL
    - 20% have max depth of THREE, of which 50% are created using GROW and 50% using FULL
    - Etc, etc until depth of 6 attained
  - I.e. 2 – 6 ramped half-half
- Node limited initialization
  - No division between FULL and GROW
  - Entirely heuristically driven

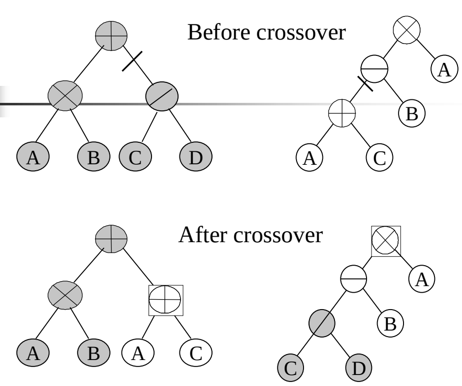
Representation – Tree Structured

- Syntactic closure necessary over any combination of argument and node
- 0-Argument – leaf nodes
  - Terminal Set
- 0 > Argument – internal nodes
  - Functional Set

Tree Structured GP – Output at Root Node



+(× A B) (/ C D)   × (- (+ A C) B) A



Parameter Selection
#1 – Variable Length, Tree

Two major numerical parameters	
Population Size (M)	Typical values 500 – 16,000
Max. Number of Generations (G)	50

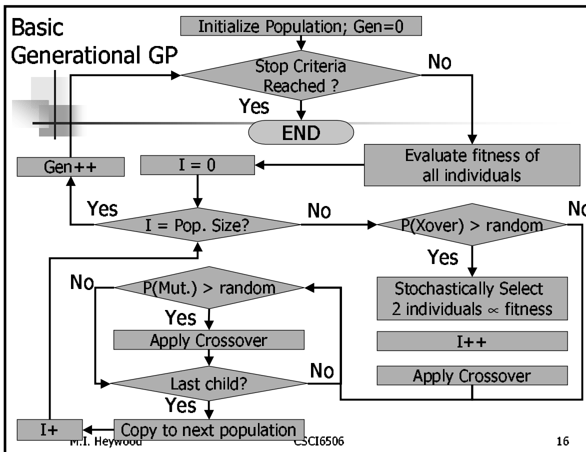
Parameter Selection
#2 – Variable Length, Tree

Minor numerical parameters	
Prob(Xover) p_c	90%
Prob(Reproduction) p_r	10%
Prob(internal Xover points) p_{ip}	90%
Max Tree Depth (D_c)	17
Max. Initial Tree Depth (D_i)	6
Prob(mutation)	0% - 10%
Elitism	0 – 2 individuals
Number of ADFs	0 – 3
If ADF, number of arguments	Problem dependent

Measuring Performance

- Computational Effort
  - Statistical model for number of generations necessary to provide solutions 99% of the time.
  - Need empirical result for one solution
- Complexity of an individual
  - Before and after simplification
- CPU time

Basic Generational GP



Fitness Revisited – Structural Complexity (representation independent)

- Biased Operators
  - Selection
    - Size a factor in selection [3]
  - Search
    - Non Destructive Crossover – child v parent [4]
    - Size Fair Crossover [5]
- Regularization
  - Parsimony Pressure [6]

Alternative GP Structures

- Linear
  - Register level transfer language
- Grammar
- Typed languages
- Recursion
- Subroutines

References

1. Koza J.R., Hierarchical Genetic Algorithms Operating on Populations of Computer Programs. Proc. 11th IJCAI, Morgan Kaufmann, pp768-774.
2. Koza J.R., Genetic Programming: On the Programming of Computers by the Means of Natural Selection. MIT Press, 1992.
3. Luke S., Panait L., Lexicographic Parsimony Pressure. GECCO-2002, Morgan Kaufmann, 829-836, 2002.
4. Soule T., Foster J.A. Removal Bias. IEEE CEC, pp 781-786, 1998.
5. Soule T. , Foster J.A. Effects of Code Growth an Parsimony Pressure on Populations in GP. Evolutionary Computation. 6(4), pp 293-309, 1998.