Introduction

Gecco 2008 Grammatical Evolution Tutorial
R. Muhammad Atif Azad and Conor Ryan
Biocomputing and Developmental Systems Group
Department of Computer Science and Information Systems
University of Limerick
July 12, 2008

Issues with GP

- Function/terminal set must have “closure”
- Single types only
- Trees grow, or “bloat”

Outline

- Introduction
- Grammatical Evolution
- Genetic Operators
- GAuGE
- Chorus
- Degeneracy
- Wrapping
- Search Techniques
- The Future
Biological Phenomena

- No simple one to one mapping
- Genes produce proteins
- Proteins combine to create phenotype
- Linear strings
  - Genomes are always held on strings
- Unconstrained search
  - Repair not performed

Grammatical Evolution

- Grammatical Evolution (GE)
  - QA to evolve programs
  - Morphogenetic Effect
  - Genotype mapped to phenotype
  - Phenotype is a compilable program
  - Genome governs mapping of a BNF/attribute grammar definition to the program

Grammatical Evolution

Here genome (a binary string) is mapped to compilable C code

- Can potentially evolve programs in any language, with arbitrary complexity
- Any structure that can be specified with a grammar, e.g. graphs, neural networks, etc.

Languages Definition

- Backus Naur Form (BNF)
  - Notation for expressing a language’s grammar as Production Rules
  - BNF Grammar consists of the tuple \(<T,N,P,S>\) where
    - T is Terminals set
    - N is Non-Terminals set
    - P is Production Rules set
    - S is Start Symbol (a member of N)
  - BNF Example
    \[ T = \{ \sin, \cos, \tan, +, -, /, *, X, (, ) \} \]
    \[ S = < expr > \]
Grammatical Evolution

Grammars

BNF Definition

N = \{ expr, op, pre_op \}

And P can be represented as:

(1) <expr> ::= <expr> <op> <expr> (A)
    | ( <expr> <op> <expr> ) (B)
    | <pre-op> ( <expr> ) (C)
    | <var> (D)

(2) <op> ::= + (A)
    | - (B)
    | / (C)
    | * (D)

A Genetic Algorithm is used to control choice of production rule

Architecture

Problem
Grammar
Search Algorithm

Program

Related GP Systems

<table>
<thead>
<tr>
<th>Name</th>
<th>Genome Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Koza</td>
<td>Tree, Direct</td>
</tr>
<tr>
<td>Banzhaf et al</td>
<td>Linear, Direct</td>
</tr>
<tr>
<td>Gruau</td>
<td>Tree, Targan Grammar</td>
</tr>
<tr>
<td>Whigham</td>
<td>Tree, Derivation Tree</td>
</tr>
<tr>
<td>Wong &amp; Leung</td>
<td>Tree, Logic Grammars</td>
</tr>
<tr>
<td>Paterson</td>
<td>Linear, Grammar</td>
</tr>
</tbody>
</table>

- Repair mechanisms:
  - Koza - none needed
  - Banzhaf - required for syntactically legal individuals
  - Gruau - none needed
  - Whigham - all crossovers subject to repair
  - Wong & Leung - all crossovers subject to repair
  - Paterson - under/overspecification
In contrast GE uses:
- BNF - Paterson/Whigham/Wong etc.
- Variable Length Linear Chromosomes - Koza/Gruau/Banzhaf
- Genomes encode pseudo-random numbers
- Degenerate Genetic Code
- Several genes map to same phenotype
- Wrap individuals
- Use 8 bit codons
- Each codon represents at least one Production Rule
- Gene contains many codons
- Pseudo-random numbers determine what production rule will be used

Expression of a Codon results in an Amino Acid (choice in the derivation sequence)
- Amino acids can combine to form a functional protein (i.e. terminals such as +, X or Sin can combine)
Grammatical Evolution

**Example Individual**

- To complete BNF definition for a function written in a subset of C we include:
  
  \[
  \text{<func>} ::= \text{<header>}
  \]
  
  \[
  \text{<header>} ::= \text{float symb(float X)} \text{ <body>}
  \]
  
  \[
  \text{<body>} ::= \text{<declarations><code><return>}
  \]
  
  \[
  \text{<declarations>} ::= \text{float a;}
  \]
  
  \[
  \text{<code>} ::= a = \text{<expr>};
  \]
  
  \[
  \text{<return>} ::= \text{return (a);}
  \]

- Note implementation details:
  
  - Function is limited to a single line of code
  
    \[
    \text{Function is limited to a single line of code}
    \]
  
  - If required can get GE to generate multi-line functions... modify
    
    \[
    \text{<code>} ::= \text{<line>;}
    \]
    \[
    | \text{<line>; <code>}
    \]

(University of Limerick)

July 12, 2008 16 / 82

---

**Example Individual**

In this subset of C all individuals of the form

\[
\text{float symb(float x)} \{
\]

\[
\text{float a;}
\]

\[
a = \text{<expr>};
\]

\[
\text{return (a);}
\]

Only \text{<expr>} will be evolved

- Each non-terminal is mapped to a terminal before any others undergo a mapping process

(University of Limerick)

July 12, 2008 17 / 82

---

**Example Individual**

- Given the individual
  
  \[
  220 \quad 203 \quad 51 \quad 123 \quad 2 \quad 45 \quad \ldots \text{what will happen?}
  \]

  - \text{<expr>} has 4 production rules to choose from:
    
    \[
    \text{(1) \quad \text{<expr>} ::= \text{<expr> <op> <expr> (A)}}
    \]
    \[
    \text{\quad | ( <expr> <op> <expr> ) (B)}}
    \]
    \[
    \text{\quad | (pre-op) ( <expr> ) (C)}}
    \]
    \[
    \text{\quad | (var) (D)}}
    \]

  - Taking first codon 220 we get
    
    \[
    \text{220 MOD 4 = 0}
    \]

    \[
    \text{Gives \text{<expr> <op> <expr>}}
    \]

  - Next choice for the first \text{<expr>}
    
    \[
    \text{Taking next codon 203 we get 203 MOD 4 = 3}
    \]

    \[
    \text{Gives \text{<pre-op> ( <expr> )}}
    \]

- \text{<var>} involves no choice
  
  - Mapped to \text{X}, only one production
  
  - Now have \text{X \text{<op> <expr>}}

  (University of Limerick)

July 12, 2008 18 / 82

---

**Example Individual**

- \text{<op> involves no choice}
  
  - Mapped to \text{X}, only one production
  
  - Now have \text{X \text{<op> <expr>}}

  (University of Limerick)

July 12, 2008 19 / 82

---

**Example Individual**

- \text{<op> involves no choice}
  
  - Mapped to \text{X}, only one production
  
  - Now have \text{X \text{<op> <expr>}}

  (University of Limerick)

July 12, 2008 20 / 82

---

2344
Codons are polymorphic

- When mapping <expr>, we calculate
  \[ 220 \mod 4 \]
- However, if we were mapping <pre - op> with 220, we would calculate
  \[ 220 \mod 3 \]
  because there are just three choices
- Meaning of a codon depends on its context

Mapping Process

- No simple one to one mapping in GE
- Mapping Process to generate programs
  - Separate Search and Solution Spaces
  - Ensure validity of individuals
  - Remove language dependency
  - Maintain diversity
Neutral Mutations
- Mutations having no effect on Phenotype Fitness
- Help preserve individual validity
- Gradual accumulation of mutations without harming functionality
- Revisit later

Initialisation
- Individuals are strings of random numbers
  - No guarantee that they will terminate
  - Individuals can be very short

Production
- No explicit start symbol
- On average, a quarter of all individuals are just one point

Sensible Initialisation
- Generate a spread of individual sizes.
  - Based on Ramped Half and Half initialisation in GP
  - For all tree depths from 2 to maximum size
  - Generate an equal number of trees of that size
  - Use full for 50% of population
  - Use grow for 50% of population
- Similar in GE, but generate derivation trees of equivalent size
**Sensible Initialisation - 2**

- Record which number choice was made for each step
- Perform an "unmod" on list of choices
  - Produce a number between 0 and 255 that produces the original number when moded by the number of choices for that production rule
- Ensures that all individuals are valid
- Reduces the number of clones (easier to detect)
- Eliminates single point individuals (if desired)

**Genetic Operators**

- Perform unconstrained Evolutionary Search
- GE employs standard operators of Genetic Algorithms
  - Point mutation, one-point crossover etc.
  - Sometimes modified version of one-point crossover, Sensible Crossover, is used:
    - Effective length
    - Actual length

**Crossover**

- What actually happens in crossover?
- Preliminary - Visualisation.

- Crossover is performed at genotypic level
Genetic Operators

Crossover

- What actually happens in crossover?
- Preliminary: Visualisation.

Crossover is performed at genotypic level.

Grammatical Evolution

July 12, 2008 30 / 82
Genetic Operators

Ripple Crossover

- Analyse 1-point crossover in terms of derivation & syntax trees
- Use a closed grammar

\[ E ::= (+ E E) \{0\} \]
\[ (- E E) \{1\} \]
\[ (\times E E) \{2\} \]
\[ (% E E) \{3\} \]
\[ X \{4\} \]
\[ Y \{5\} \]

- No polymorphism, because there is only one non-terminal, i.e. one context

Different Views of Crossover
Genetic Operators

Ripple Crossover

Different Views of Crossover

Genetic Operators

Ripple Crossover

Rebuilding individuals

- Parent left with "spine"
  \[ \begin{array}{c}
  \text{E} \\
  \text{E} \\
  \text{E} \\
  \text{E} \\
  \text{E} \\
  \end{array} \]

- Tail swapped with other parent
  \[ \begin{array}{c}
  4 \ 5 \ 9 \ 2 \ 5 \ 2 \ 2 \\
  \end{array} \]

- Unmapped E terms must be mapped
  - Use tail from other parent

Intrinsic Polymorphism

- With more than one non-terminal, a codon could be used differently in the offspring
  \[ \begin{array}{c}
  1 \ 0 \ 0 \ 2 \ 0 \ 1 \\
  1 \ 0 \ 0 \ 2 \ 0 \ 1 \\
  1 \ 0 \ 0 \ 2 \ 0 \ 1 \\
  \end{array} \]

expr ::= var | expr op expr
opr ::= + | * | - | %
var ::= x | y
Genetic Operators

Ripple Crossover

Intrinsic Polymorphism

With more than one non-terminal, a codon could be used differently in the offspring

\[ \begin{matrix}
1 & 0 & 0 & 2 & 0 & 1 \\
1 & 0 & 0 & 2 & 0 & 1 \\
1 & 0 & 0 & 2 & 0 & 1 
\end{matrix} \]

Grammar Evolution

July 12, 2008 34 / 82

Effects of Ripple Crossover

Symbolic Regression Grammars

Closed Grammar

\[ E ::= x \]
\[ \mid \overset{=} \cdot E \mid \overset{=} + E \]
\[ \mid \overset{=} - E \mid \overset{=} / E \]

And the context-free grammar:

\[ \text{Exp} ::= \text{Var} \mid \text{Exp Op Exp} \]
\[ \text{Var} ::= x \]
\[ \text{Op} ::= + \mid - \mid / \]

Grammar Evolution

July 12, 2008 35 / 82
Genetic Operators Ripple Crossover

Santa Fe ant trail grammars
Closed grammar:
\[ E ::= \text{move()} | \text{left()} | \text{right()} | \text{iffoodahead}(E, E) | \text{progl}(E, E) \]
Context free grammar:
\[ \text{Code} ::= \text{Line} | \text{progl}(	ext{Line}, \text{Code}) \]
\[ \text{Line} ::= \text{Condition} \mid \text{Action} \]
\[ \text{Action} ::= \text{move()} | \text{right()} | \text{left()} \]
\[ \text{Condition} ::= \text{iffoodahead}(\text{Code}, \text{Code}) \]

Symbolic Regression Success Rates
Both ripple crossovers start more slowly, but reach higher fitness.

Santa Fe Success Rates
Both ripple crossovers again start more slowly, but reach similar fitness.

Santa Fe - Extended Run
Success rates on the Santa Fe ant trail problem, averaged over 100 runs, for 250 generations. Ripple crossovers start slowly, but reach higher fitness.
Genetic Operators

Alternative Crossovers

Other types of Crossover?

- Homologous Crossover
  - Try not to cross in identical areas
  - Uniform

- Same size homologous
- Same size two point

Homologous Crossover - First point

- Record rule histories for each individual
- Align rule histories of parents

Homologous Crossover - Second Point

- Choose second point outside of area of similarity
- Align rule histories of parents
Genetic Operators

Alternative Crossovers

Crossover comparisons (Cumulative Freq. Success)

Productivity of Operators (Ratio of successes)

Relative size of crossover fragments

Headless Chicken - Crossover or Macromutation

- Appears Crossover works
- 50% material exchange with 1-point over entire runs
- If useful material exchanged then swapping random fragments should degrade performance?
Genetic Operators

Alternative Crossovers

Headless Chicken Comparison

Why does crossover work?
- Take a cue from GP crossover - The “Eve” Effect:
  - All individuals in the final generation tend to evolve from the same ancestor
  - The upper parts of individuals tend to come from the same individual

GE View of Eve Effect?
- Individuals grow from left to right
- Size of region of similarity increases over time
- Area immediately beyond region of similarity is “region of discovery”:
  - Region of Region of Similarity Discovery

Size of region of similarity increases over time
- Decreasing prob. of identical decoded values
Genetic Operators
Explanation:

Size of region of similarity increases over time
- Area immediately beyond region of similarity is "region of discovery":

GAuGE
Introduction
Genetic Algorithms using Grammatical Evolution

Purpose:
- Position independent genetic algorithm;
- No under- or over-specification;
- Independent of search engine.

Based on mapping process (similar to GE):
- Specify position and value of each variable at genotypic level;
- Map genotype strings into functional phenotype strings.

Mapping in GAuGE System
Transform binary string into integer string:
- Problem has 4 variables ($\ell = 4$), with range 0...7;
- Choose position field size ($pfs = 2$);
- Choose value field size ($vfs = 4$);
- Calculate binary string length:

$$L = (pfs + vfs) \times \ell = (2 + 4) \times 4 = 24 \text{ bits}$$
Transform binary string into integer string:
- Problem has 4 variables ($\ell = 4$), with range 0...7;
- Choose position field size ($pfs = 2$);
- Choose value field size ($vfs = 4$);
- Calculate binary string length:

$$L = (pfs + vfs) \times \ell = (2 + 4) \times 4 = 24 \text{ bits}$$

001001101101010100010010
Binary string

0123
Phenotype

Calculating Phenotype
Where is Gauge useful?

- GAUGE adapts the representation to the problem
- Useful where interactions between genes not known
- GAUGE is cheap
- Far less complicated than algorithms that try to model gene interactions/relationships
- GAUGE discovers saliency
  - Most important genes end up on left side of strings

Chorus

- Mapping Independent Codons - no ripple effect
- Codon % Total number of rules in the grammar
- Competition between the Genes
- Concentration Table
- Variable length binary strings
- 8 bit codons

Grammar specification

\[
S = \langle expr \rangle \\
(0) \langle expr \rangle ::= \langle expr \rangle \ \langle op \rangle \ \langle expr \rangle \\
(1) \ | \ ( \langle expr \rangle \ \langle op \rangle \ \langle expr \rangle \ ) \\
(2) \ | \ (pre-op) \ | \ \langle expr \rangle \\
(3) \ | \ \langle var \rangle \\
(4) \ \langle expr \rangle ::= \ + \\
(5) \ | \ - \\
(6) \ | \ \times \\
(7) \ | \ \div \\
(8) \ | \ \langle pre-op \rangle ::= \ \text{Sin} \\
(9) \ | \ \text{Cos} \\
(A) \ | \ \text{Exp} \\
(B) \ | \ \text{Log} \\
(C) \ | \ \langle var \rangle ::= \ 1.0 \\
(0) \ | \ X
\]
Four non-terminals:

- `<expr>` 0..3, `<op>` 4..7, `<pre-op>` 8..B, `<var>` C..D

Cumulative Freq. with and without degeneracy

- No huge difference - Normal, 4- and 6-bit top three in both

Degeneracy

Performance

Cumulative Freq. with and without degeneracy

Mean Variety - Any degeneracy helps!
Degeneracy
Variety

Conclusions

- Improves genetic diversity
- Improves frequency of success on Santa Fe ant trail
- Tunable/Evolvable Degeneracy a good idea?

Figure: The number of invalid individuals for each generation in the presence and absence of wrapping.

Figure: Number of individuals wrapped on the symbolic regression and Santa Fe trail problems.

Number of individuals wrapped

- Wrap Count & Invalid Individuals

Figure: Number of individuals wrapped on the symbolic regression and Santa Fe trail problems.

Wrapping and Invalid Individuals
**Performance**

- Freq. of Success

![Performance Graph](image)

Figure: Figure shows the cumulative frequency of success measures on both problems with and without the presence of wrapping.

**Genome Lengths**

- Actual length
  - Entire length of individual
- Effective length
  - Number of codons used
- (Note: Can be less than or greater than actual length)

**Summary**

- For SR (left) wrapping off has the longest actual length
- Effective length virtually the same
- For SF (right) wrapping on longer in both cases.

Conclusions:
- Wrapping improves frequency of success on Santa Fe ant trail
- No effect on Symbolic Regression cumulative frequency
- Provides some constraint on genome lengths
Wrapping & Degeneracy

- Removing both...
  - Cumulative frequency of success degrades
  - Genome lengths increase over 60% on Symbolic Regression
  - Genetic diversity no worse than without degeneracy alone

Search Techniques

- Other techniques
  - Simulated Annealing
  - Hill Climbing
  - Random Search

Comparison

- Three standard GP problems
  - Santa Fe trail
  - Symbolic Integration \(\int \cos(x) + 2x + 1\)
  - Symbolic regression \(x^2 + x^3 + x^4 + x\)

<table>
<thead>
<tr>
<th>Metaheuristic</th>
<th>RS</th>
<th>HC</th>
<th>SA</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Santa Fe trail</td>
<td>14%</td>
<td>9%</td>
<td>6%</td>
<td>14%</td>
</tr>
<tr>
<td>Symbolic Integration</td>
<td>3%</td>
<td>3%</td>
<td>3%</td>
<td>100%</td>
</tr>
<tr>
<td>Symbolic Regression</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>59%</td>
</tr>
</tbody>
</table>

The Future

- Evolving machine code (Machine Code Grammatical Evolution - MCGE)
- The Grammar (Attribute Grammars)
- Search & Evolutionary Dynamics
- Applications
- Newest Code Release
- http://www.grammaticalevolution.org/libGE
Opportunities

- Programmer
  - EUR 31,000 - EUR 33,000
  - Possible to register for part time PhD
- PhD Students
  - EUR 24,000