Automatic Algorithm Invention

by Wes Faler of Part-Time Scientists for 28C3, Berlin Germany, 2011
wf@ptscientists.com
• Why
• What
• How
• Code and Details
• Scaling
• State
• Your turn!
• Part Time Scientists needs a De-Bayer algorithm for our images.
What - Bayer Image Filter

Source: Wikimedia
• Given:
  ◦ An existing system structure
    • Inputs
    • Output(s)
    • Formula structure
  ◦ Test cases
  ◦ Constraints
• Create:
  ◦ The optimal set of parameters

\[ a \text{ Red} + b \text{ Blue} + c \text{ Green}_0 + d \text{ Green}_1 \]
Given:
- A partial existing system structure
  - Inputs
  - Output(s)
  - Formula structure
- Test cases
- Constraints

Create:
- The optimal set of parameters
- An equation or algorithm
Invention - GP

\[ (2.2 - \left(\frac{X}{11}\right)) + (7 \times \cos(Y)) \]
• Cartesian Genetic Programming (CGP)
  ◦ Generates equations like circuits.
  ◦ Parallelizable results.
  ◦ FPGA friendly.
• Make a random “circuit”.
• Grid layout.

How – CGP
Score the circuit.

Test #1: Terrible
Test #2: Terrible
Test #3: Terrible

Score: Terrible*3
• Make random changes and rescore.

Test #1: Terrible
Test #2: Terrible-4
Test #3: Terrible

Score: Terrible*3-4
• Start with 1 parent “circuit”.

500

How - CGP
• Start with 1 parent “circuit”.
• Make mutant children.
- Start with 1 parent “circuit”.
- Make mutant children.
- Score everyone.
• Start with 1 parent “circuit”.
• Make mutant children.
• Score everyone.
• Promote the best child that isn’t worse than the parent.

How - CGP
• Start with 1 parent “circuit”.
• Make mutant children.
• Score everyone.
• Promote the best child that isn’t worse than the parent.
Neutral Search - Scores

New < Old

New <= Old
Neutral Search – Mutations

New < Old

New <= Old
**Terminology**

- $N_i$: Input indexes
- $N_c$: Constant
- $N_r$: Operator type
- $N_o$: Input indexes

**Gene**

- Operator type
- Constant
- Input indexes

*Image of a diagram showing the flow of data through a gene with operator types and input indexes.*
• Individual
  ◦ Representation
  ◦ Random creation
• Execution
<table>
<thead>
<tr>
<th>$N_i$</th>
<th>$N_o$</th>
<th>Fitness ($N_P$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Testing one test case

Typical vector operation:
Fitness = |actual − expected|^2
Finish the loop

\[ \sum \sum \sum \sum \]

Sum fitness
Select best
Promote
Mutate

\( N_i \quad N_o \quad \text{Fitness (} N_P \text{)} \)
• Test cases
• Population
• Mutator
Demo Explained

Test cases

Validation Image

Bayer Image
CGP-based De-Bayer filter
Scaling

$O(N_o^{2+})$
Scaling – Simple Chromosomes
Scaling – Complex Chromosomes
Beyond functions - State

State data saved for next run.
class Widget {
private:
    float fState[2];
    float gState[2];
    float sharedState[4];
public:
    void f(const float input[7],
            float output[3]);
    void g(const float input[4],
            float output[1]);
};

2 chromosomes with private and shared state
• Visualize!
• Tests should give partial success
• Increase population
• Keep mutation rate low enough
- CGP on GPU
  - Especially validation!
- Islands
- More data visualizations
- Parameter tuning with PSO
- Open source
- Danke!

- <shameless recruiting ad>
  ◦ Join us at ptscientists.com!
- </shameless recruiting ad>

- Wes Faler of Part-Time Scientists
  ◦ wf@ptscientists.com
• Julian Miller (inventor of CGP)
  ◦ http://sites.google.com/site/julianfrancismiller/professional

• CGP Book
  ◦ “Cartesian Genetic Programming”
  ◦ http://www.springer.com/computer/theoretical+computer+science/book/978-3-642-17309-7

• “Evolved to Win” e-book
  ◦ http://www.moshesipper.com/etw/