William B. Langdon and Oliver Krauss

Genetic Improvement of Data for Maths Functions

Online 10-14. July 2021
Genetic Improvement of data

– Optimize constants, i.e. data
– Maintain software
– Evolve new or better functionality
– Different type of Genetic Improvement
Why is this relevant?

Figure: 1,202,711 integer constants in GNU C library
Evolved functions

<table>
<thead>
<tr>
<th>Start</th>
<th>Evolved</th>
<th>Accuracy</th>
<th>secs</th>
</tr>
</thead>
<tbody>
<tr>
<td>sqrt</td>
<td>cbrt()</td>
<td>$\sqrt[3]{x}$</td>
<td>dp i.e. $\leq 6.7 \times 10^{-16}$</td>
</tr>
<tr>
<td>sqrt</td>
<td>log2()</td>
<td>$\log_2 x$</td>
<td>dp i.e. $\leq 2.2 \times 10^{-16}$</td>
</tr>
<tr>
<td>sqrt</td>
<td>invsqrt()</td>
<td>$x^{-1/2}$</td>
<td>dp i.e. $\leq 2.2 \times 10^{-16}$</td>
</tr>
<tr>
<td>sqrt</td>
<td>reciproc()</td>
<td>$x^{-1}$</td>
<td>dp i.e. $\leq 2.2 \times 10^{-16}$</td>
</tr>
</tbody>
</table>

*dp = double precision accuracy
How math functions work I

Figure: Newton Raphson Approximation

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$
How math functions work II

\[ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \]

- load double precision value
- select initial guess from lookup table

Figure: Newton Raphson with Lookup Table
Evolving cube root from square root I

- Manual modification of glibc sqrt
- Covariance matrix adaption evolution strategy (CMA-ES)
  - For each *bin* in the lookup table
  - Fitness is *result cubed*
  - Random tests of several thousand double precision numbers
Figure: Fitness Landscape for cube root in GI (smaller is better)
Results

Table: Accuracy and total time time (seconds) for CMA-ES

<table>
<thead>
<tr>
<th>Start</th>
<th>Evolved</th>
<th>Accuracy</th>
<th>secs</th>
</tr>
</thead>
<tbody>
<tr>
<td>sqrt → cb()</td>
<td>$\sqrt[3]{x}$</td>
<td>dp i.e. $\leq 6.7 \times 10^{-16}$*</td>
<td>270</td>
</tr>
<tr>
<td>sqrt → log2()</td>
<td>$\log_2 x$</td>
<td>dp i.e. $\leq 2.2 \times 10^{-16}$</td>
<td>6</td>
</tr>
<tr>
<td>sqrt → invsqrt()</td>
<td>$x^{-1/2}$</td>
<td>dp i.e. $\leq 2.2 \times 10^{-16}$</td>
<td>6</td>
</tr>
<tr>
<td>sqrt → reciprocol</td>
<td>$x^{-1}$</td>
<td>dp i.e. $\leq 2.2 \times 10^{-16}$</td>
<td>6</td>
</tr>
</tbody>
</table>

*Accuracy better than C++ and Java implementations. Runtime faster than Java implementation [1]
Conclusion

- Software can be maintained via GI
- Low effort
  - Takes just a few seconds
  - Source code and test case already exist
- Small changes
  - modifications comprehensible
  - higher acceptance by developers?
- Try it yourself!

Further information available at upcoming ACM TELO publication
http://www0.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/Langdon_TELO.pdf [2]. Replication package on GitHub
https://github.com/oliver-krauss/Replication_GI_Division_Free_Division
Contact

Bill Langdon
w.langdon@cs.ucl.ac.uk

Oliver Krauss
oliver.krauss@fh-hagenberg.at
Bibliography 1
