Evolving Power-Efficient RNGs

David R. White
University of Glasgow

39th CREST Open Workshop
Tuesday 24th February 2015
Table of Contents

Introduction

Evolving RNGs with GP

Measuring Power Consumption and Functionality

Experimental Method

Results

Conclusions and Postscript
Table of Contents

Introduction

Evolving RNGs with GP

Measuring Power Consumption and Functionality

Experimental Method

Results

Conclusions and Postscript
This work was my first paper, at GECCO 2008.

Followed by work on non-functional properties with GP and a collaboration with Andrea Arcuri.

Some of the earliest work on using GP to optimise non-functional properties.
Resource Efficient Software: Target Platforms

WSN Motes
- Hardware limited (e.g. 64KB, 12MHz Mote)
- Multiple, demanding, requirements
- Interdependence between requirements
- What happens when you just don’t have enough resources?

RFID Tags
The Difficult of Manual Power Optimisation

AKA “the loneliness of the long-distance embedded systems programmer.”

“The internals of a modern processor are complete chaos”.

Anonymous RTS Professor
Standard Programming

Programmer → Compiler → Binary
Standard Programming: Limitations

Programmer → Compiler → Binary

- No automated improvement at this level
- Compiler limited by programmer's decisions
- Platform and usage almost incidental

David R. White, University of Glasgow

Evolving Power-Efficient RNGs
Standard Programming: Limitations

- No automated improvement at this level
- Compiler limited by programmer's decisions
- Platform and usage almost incidental
- Complex relationship between source and binary
Table of Contents

Introduction

Evolving RNGs with GP

Measuring Power Consumption and Functionality

Experimental Method

Results

Conclusions and Postscript
Why RNGs?

A common component; small and frequently executed.

We were interested in trade-offs, not solely optimisation.
Lamenca-Martinez et al. [2006] evolved PRNGs with a view to efficiency, building on work by Hernandez et al. [2004].

Tried to ensure efficiency through choice of function set.

See also Koza [1991], Jannink [1994], Sipper and Tomassini [1996].
Quality was measured using the Strict Avalanche Criterion as per Lamenca-Martinez et al. [2006].

Power efficiency was measured using the Sim-Wattch Simulator [Brooks et al., 2000].
Cycle-level power simulator based on SimpleScalar simulator.

Estimates power usage through approximation for each logic unit:

\[ P_d = CV_{dd}^2 af \]  \hspace{1cm} (1)

\( C \) is the load capacitance; \( V_{dd} \) the supply voltage, \( f \) the clock frequency and \( a \) an activity that estimates the amount of transistor switching.

Partly estimated, validation given by Brooks et al. [2000].
Simulation is (allegedly) inaccurate and expensive.

For our purposes it only has to be relatively accurate, most of the time.

We reduced the sample size compared to previous work, to increase speed.

Could we have just used execution time as a proxy?
Measuring Randomness: Strict Avalanche Criterion

Trying to create a function \( r(a_0 \ldots a_7) \)

Maximise nonlinearity of this function:

1. Create a random input \( a_0 \ldots a_7 \)
2. Evaluate \( o_1 = r(a_0 \ldots a_7) \)
3. Flip one bit of input to create \( b_0 \ldots b_7 \)
4. Evaluate \( o_2 = r(b_0 \ldots b_7) \)
5. Record hamming distance \( H(o_1, o_2) \)
Compare the distribution of bit flips to the binomial distribution:

$$\sum_{i=0}^{n} \frac{(C_i - E_i)^2}{E_i}$$
Our Objective Space

![Graph showing Resource (Power) vs. Error (SAC)]
Table of Contents

Introduction

Evolving RNGs with GP

Measuring Power Consumption and Functionality

Experimental Method

Results

Conclusions and Postscript
Replicated work by Hernandez et al. [2004] and achieved similar results.

Used reduced sample size.

Searched using ECJ and SPEA2.
Variance of Fitness Measures

Repeated for other programs and power consumption measure.
Experimental Framework

- ECJ
- GP & SPEA2
- Raw Fitness Values
- PRNG source
- Test Harness source
- Mersenne Twister source
- GCC Object Code
- Sim-Wattch
- Power Statistics
- Test Harness Output
- Raw Fitness Values

David R. White
University of Glasgow
Evolving Power-Efficient RNGs
Parameter Settings

| Function set | MULT, AND, SUM, NOT, OR, XOR, Logical Shift Left (LSL), LSR, Circular Shift Left (CSL), CSR |

Parameter settings left to their defaults in ECJ, as specified by koza.params.

All non-default settings taken from previous work, except for MOO-specific settings.
Table of Contents

Introduction

Evolving RNGs with GP

Measuring Power Consumption and Functionality

Experimental Method

Results

Conclusions and Postscript
Evolution of the Pareto Front

Animation
int genRand (int a0, int a1, int a2, int a3, int a4, int a5, int a6, int a7) {
    return ¬(¬(1997453768ul));
}
A Middle Ranking Low Power RNG

\begin{verbatim}
int genRand (int a0, int a1, int a2, int a3, int a4, int a5, int a6, int a7) {
    return a2 ∨ ¬(((a2 + a0) * ((a4 ⊕ ((a6 + a5) ⊕ a7)) + (a1 ⊕ a3))) ∧ ¬(¬(1997453768ul)));
}
\end{verbatim}
int genRand (int a0, int a1, int a2, int a3, int a4, int a5, int a6, int a7) {
    (2307363674ul ⊕ (a2 * a6)) + (((¬(((((((a2 * a6) ⊕ (a7 ⊕ a1)) * (a0 ⊕ a3)) ⊕ (a2 * a6)) + (a5 * a4)) >> 2307363674ul) ⊕ (a0 ⊕ a3))) + ¬((a5 * a4) * ((2307363674ul ⊕ (a7 ⊕ a1)) + (a0 ⊕ a3)))));
}
The Impact of the Mult Function

\[ \log_{10}(X^2 \text{ Fitness}) \]

- Multiply Included
- Multiply Omitted
This is about more than straightforward optimisation.

Multiple solutions vs a single, tunable, scalable algorithm.

More radically, consider manipulation rather than optimisation.
Postscript: Time Avalanche Criterion

See White et al. [2010]
Acknowledgements

My coauthors John Clark, Jeremy Jacob and Simon Poulding.

Andrea Arcuri for his collaboration.

Juan E. Tapiador and Julio Cesar Hernandez-Castro, for inspiration and creative discussion.

david.r.white@glasgow.ac.uk
Genetic Improvement Workshop
(at GECCO 2015)

Submission deadline: April 3, 2015
geneticimprovement2015.com


