Long-Term Evolution Experiment with Genetic Programming [Hot of the Press]*

William B. Langdon
Computer Science, University College London
London, UK

Wolfgang Banzhaf
Department of Computer Science and Engineering
Michigan State University, East Lansing, USA

ABSTRACT

We evolve floating point Sextic polynomial populations of genetic programming binary trees for up to a million generations. We observe continued innovation but this is limited by their *depth* and suggest deep expressions are resilient to learning as they disperse information, impeding evolvability and the adaptation of highly nested organisms and instead we argue for open complexity. Programs with more than 2 000 000 000 instructions (depth 20 000) are created by crossover. To support unbounded long-term evolution experiments LTEE in GP we use incremental fitness evaluation and both SIMD parallel AVX 512 bit instructions and 16 threads to yield performance *equivalent* of up to 1.1 trillion GP operations per second, 1.1 tera-GPops, on an Intel Xeon Gold 6136 CPU 3.00GHz server.

KEYWORDS

genetic programming, Information Theory limit on complexity, Long-Term Evolution Experiment LTEE, extended unlimited evolution, Open Complexity, Speedup technique

ACM Reference Format:
William B. Langdon and Wolfgang Banzhaf . 2022. Long-Term Evolution Experiment with Genetic Programming [Hot of the Press]. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3520304.3534065

INTRODUCTION

Since 1988 Richard Lenski and his collaborators have evolved *E. coli* bacterial for more than 75 000 generations with no end to fitness improvement in sight [2].

In [1] we describe our own Long-Term Evolution Experiment (LTEE) replacing bacteria with computer based artificial evolution by running tree based genetic programming (GP) with only crossover and without constraints for tens of thousands, even a million generations. We see in floating point GP continual innovation and improvement in fitness like in the bacteria experiments. Figure 1 shows that although the rate of innovation falls, typically better solutions are found even towards the end of the runs. There are several hundred or even a few thousand of subtree crossovers of evolved parents which give a better child.

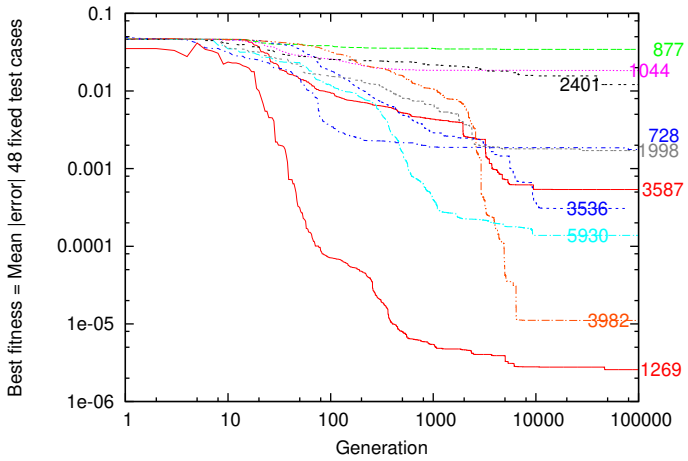

Figure 1: Evolution of mean absolute error in ten runs of Sextic polynomial with population of 500. Runs to 100 000 generations (2 stopped early). Labels give number of generations when fitness got better.

Section 4 in [1] describes the evolution of innovation particularly wrt size and depth. This is followed by a short discussion about continuous evolution (Section 5) and Section 6 discusses the implications for the evolution of open ended complexity. We conclude that even something as simple as digital evolution using GP permits continuous innovation but information theory suggests deeply nested structures slow evolution.

EXPERIMENTS

We ran three sets of experiments. The first set uses a population of 4000, the second 500 and the last 48 but for reasons of space we only report runs with 500 trees here.

In all cases evolution continued to make progress and each GP run found between 728 and 5930 fitness improvements (see Figure 1). The populations start to converge in later generations, see e.g. Figure 2 (next page).

Most improvements occurred in trees of depth between 244 and 575. However, most trees have depths between 3871 and 15 336. That is, as with the larger population, there is more innovation in shallower trees. Nonetheless GP continues to find better programs, see Figure 1, even though the population depth continues to grow. (Depth is approximately $\sqrt{2\pi \times \text{size}}$.)

All the runs with populations of 500 trees showed some cases of complete fitness convergence. That is, at some point everyone in the population had identical fitness. For example in the first run,

*We summarise our Artificial Life Journal article [1].

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3534065

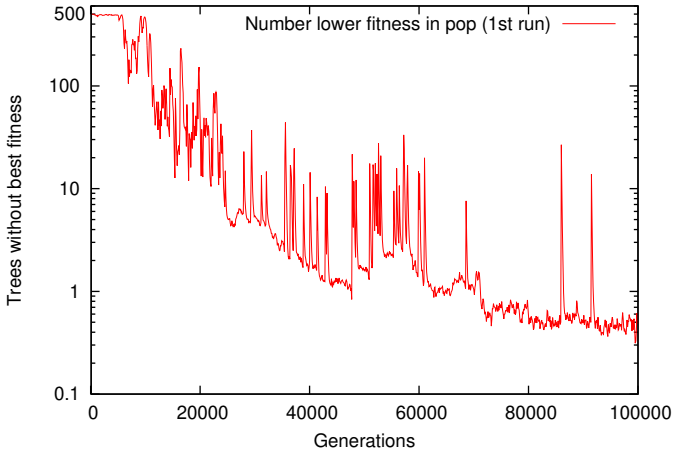

Figure 2: Fitness convergence in example Sextic polynomial pop=500 run, up to 100 000 generations (smoothed by plotting running average over 100 generations). Across the whole of each run and across the ten runs in almost half the generations the whole population has identical fitness. (The means for each individual run are between 25% and 63%.)

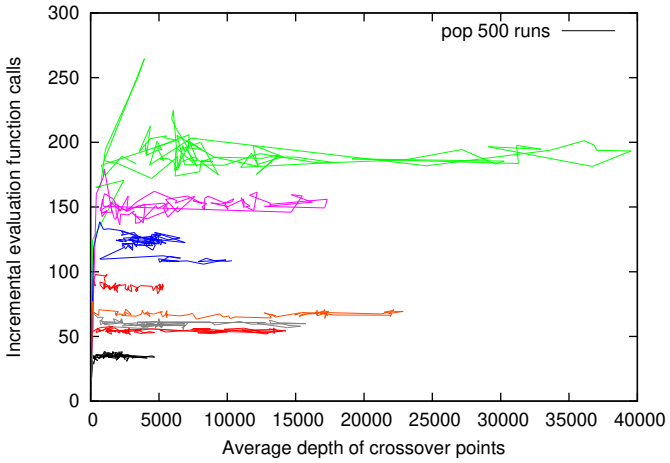

Figure 3: Mean number of nested functions disrupted by crossover, i.e. levels above XO point. Note stability even though depth of XO increase enormously. (Data collected every 1000 generations for eight runs.) Colours as Figure 1.

the whole population has identical fitness 33 143 times, 30% of the run. (See also Figure 2.)

Figure 3 plots the evolution of the mean number of nested function levels where run time fitness evaluation is changed by crossover on any of the 48 test cases. Each generation the evaluation on all the test cases of almost all the code in the population is unchanged.

IS THERE A LIMIT TO EVOLUTION?

In the floating point experiments, except with tiny populations (48), there is no hint of either evolution of fitness or bloat stopping dead.

In the binary 6-Mux Boolean problem there are only 65 different fitness values. Therefore the number of fitness improvements is very limited. And an end to bloat was found. By which we mean it was possible for trees to grow so large that crossover was unable to disrupt the important part of their calculation next to the root node and many generations were evolved where everyone had identical fitness. This led to random selection and random fluctuations in tree size, i.e. enormous trees but without a tendency for progressive endless growth. This did not happen here. Even with populations containing Sextic polynomial trees of hundreds of millions of nodes, crossover can sometimes still be disruptive and this is sufficient to drive tree size to increase.

Can bloat continue forever? Section 5 in [1] presents arguments for and against.

EVOLUTION OF OPEN COMPLEXITY

The success of incremental evaluation has some profound implications for the evolution of complex programs. It shows the progressive concealing of large effects by long chains of computation. That is, a deep crossover, mutation, run time perturbation, glitch, error, etc., etc., usually has no impact.

From the point of view of evolving complex systems, it is not sufficient for an organism to be large. Indeed, instead of deep complexity, we shall need complex systems to be open, allowing changes to percolate out to their environment. We might want such a system to be "lung like", with many passageways, allowing the run time impact of mutational changes ready access to the outside. Similarly we might view the evolving organism as a "small world" network which permits the impact of code updates or training events to pass through only a small number of nodes before reaching an external view point. Unless there are short paths, the system risks becoming onion-like: where perhaps inner layers once evolved but are now encased in newer outer layers and now adaptation occurs only in the outermost layers. Indeed, the inner layers could congeal into a static lifeless mass of fossilised code like a planetary core, with learning only occurring on the outermost crust.

CONCLUSIONS

Evolving Sextic polynomial trees for up to a million generations, during which some programs grow to two billion nodes, suggests even a simple genetic programming (GP) floating point benchmark allows long-term fitness improvement over thousands of generations.

However without short cuts, highly nested routines are robust and resistant to innermost changes. And so we suggest the opposite: large evolvable organisms will have to be open complex systems with many short paths rapidly connecting some of the learning, adaptation or mutation sites to the environment.

Acknowledgements. This work was inspired by conversations at Dagstuhl Seminar 18052 on Genetic Improvement of Software.

Parallel GPquick code is available via `http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/gp-code/GPinc.tar.gz`

References

[1] William B. Langdon and Wolfgang Banzhaf. 2022. Long-Term Evolution Experiment with Genetic Programming. *Artificial Life* 28, 2 (2022).

[2] Richard E. Lenski et al. 2015. Sustained fitness gains and variability in fitness trajectories in the long-term evolution experiment with Escherichia coli. *Proc. R. Soc. B: Biol. Sci.* 282, 1821 (22 Dec 2015). http://dx.doi.org/10.1098/rspb.2015.2292