14th Belgian-Dutch Conference on Artificial Intelligence (BNAIC'02) 21-22 Oct 2002, Leuven, Belgium, Hendrik Blockeel *(ed.)*

# Size of Random Programs to ensure Uniformity

W. B. Langdon
Computer Science, University College, London,
W.Langdon@cs.ucl.ac.uk
http://www.cs.ucl.ac.uk/staff/W.Langdon

**Abstract**

Fitness distributions (landscapes) of programs tend to a limit as they get bigger. Markov chain convergence theorems give general upper bounds on the linear program sizes needed for convergence. Tight bounds (exponential in $N$, $N \log N$ and smaller) are given in [1] for the outputs of five computer models (any, average, cyclic, bit flip and Boolean). Mutation randomises a genetic algorithm population in $\frac{1}{4}(l+1)(\log(l)+4)$ generations. While [2] considers convergence of functions. We restate the results $\frac{1}{2}N(\log(m)+4)$ and $\mathrm{O}(N)$–$\mathrm{O}(N^{3/2})$ for a genetic programming (GP) like model.

If we generated a large number of random programs and measured their characteristics (such as the value they output) this random sample would approximate the characteristics of all programs (Monte Carlo sampling). Instead of actually explicitly generating all these programs, we can apply Markov theory of random processes to random programs, to get properties of the random program and therefore bulk properties of all programs.

Using this we proved that the fitness distribution of sufficiently large programs will eventually converge [3]. However Markov chain theory can also be used to give numeric estimates of the length of random linear genetic programs, such that the distribution of their *outputs* is independent of their size. The size depend heavily on the type of computer, the fitness function, and scales with the size of the computer's memory [1]. We have also considered how big programs must be to ensure the proportion of *functions* they implement is close to the limit [2].

## Convergence of Boolean Computer Programs

Assume the CPU has 4 Boolean instructions: AND, NAND, OR and NOR. In each operation, two bits of data are read from the $N$ bits of memory. The Boolean operation is performed on the two bits, a one bit answer is created and stored in memory. A random program is simply a sequence of random instructions. I.e. two random input memory locations, a random choice of one of the four instructions and a random destination for the output bit.

Note that the CPU is as likely to generate a 0 as a 1. That is, each instruction has a 50% chance of inverting exactly one bit (chosen uniformly) from the memory and a 50% chance of doing nothing. Eventually each bit will be equally likely to be 0 or 1. Using the convergence analysis given in [4, pages 28–30] and [5] yields $||\mu_l - \pi||^2 \leq \frac{1}{2}(e^{Ne^{-\frac{2}{N}l}} - 1)$. Requiring the difference $||\mu_l - \pi||$ between

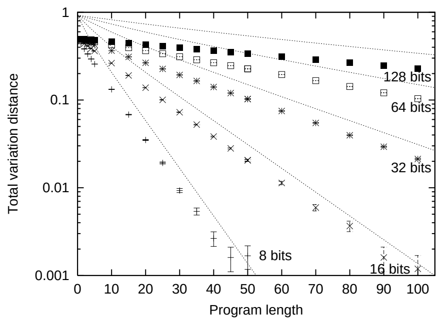


Figure 1: Convergence of outputs of random 3 bit Boolean (AND, NAND, OR, NOR) linear programs with different memory sizes. Note the agreement with upper bound $\sqrt{1/2(\exp(me^{-2l/N}) - 1)}$

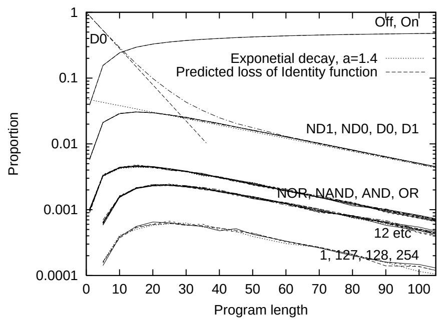


Figure 2: Convergence of Boolean functions, as Figure 1 (8 bits). Almost all programs implement Always-on or Always-off. Short programs are more varied and many implement the Identity function "D0".

the probability distribution of outputs of program of length $l$, $\mu_l$, and the uniform limiting distribution, $\pi = 2^{-N}$, not to exceed 10% gives the upper bound $l \leq \frac{1}{2}N(\log(N)+4)$. Considering only $m$ output bits gives $l \leq \frac{1}{2}N(\log(m)+4)$. Figure 1 confirms this.

The distribution of functions converges ($l \gg N/m$). In the limit, each of the $2^m$ possible constant functions are equally likely, and the proportion of each the other $2^{m\times 2^n}$ functions falls exponentially at the same rate, $\mathrm{O}(l^{-\sqrt{2/aN^3}})$. See Figure 2. Write protecting the inputs ensures non-trivial functions survive in the limit of long programs [2].

Once the distribution of functions has converged, the distribution of program fitness' must also converge, but it might only need much shorter programs.

# References

[1] W. B. Langdon. Convergence rates for the distribution of program outputs. In W. B. Langdon *etal.*, editors, *GECCO-2002*. Morgan Kaufmann.

[2] W. B. Langdon. The distribution of boolean functions. In J. Rowe *etal.*, editors, *Foundations of Genetic Algorithms VII*, 2002.

[3] W. B. Langdon and Riccardo Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2002.

[4] Persi Diaconis. *Group Representations in Probability and Statistics*, Institute of Mathematical Sciences, Hayward, California, 1988.

[5] Jeffrey S. Rosenthal. Convergence rates for Markov chains. *SIAM Review*, 37(3):387–405, 1995.