The Grand Challenge in Non-Classical Computation: International Workshop, Stepney and Emmott, eds, York, 18-19 Apr 2005

The Distribution of Amorphous Computer Outputs

W. B. Langdon
http://www.cs.essex.ac.uk/staff/W.Langdon
Computer Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK

Abstract

Fitness distributions (landscapes) of programs tend to a limit as they get bigger. Markov minorization gives upper bounds $((15.3 + 2.30m)/\log I)$ on the length of program run on random or average computing devices. I is the size of the instruction set and m size of output register. Almost all programs are constants. Convergence is exponential with 90% of programs of length $1.6\ n2^N$ yielding constants (n = size input register and size of memory = N). This is supported by experiment.

1 The Amorphous or Average Computer

In Computer Science we are used to the notion that computers are highly designed, precision engineered artifacts. Nevertheless we can theoretically analyse more amorphous computing devices. Indeed nanotechnology may be a route to their practical construction and use. Consider an abstract machine whose instruction set, rather than being designed, is chosen at random. We can consider random linear computer programs as Markov processes which move the computer from one state to another [1].

1.1 Convergence of Outputs

We start by considering what happens when a single instruction is executed. Then consider two consecutive instructions, then a program of a instructions and so on.

Suppose given any possible data in memory each of the I instructions independently randomises it. Assume the random computer also has N bits of memory. So there are 2^N possible memory patterns. Therefore the computer is always in one of 2^N states. Since the number of states is much bigger than the number of instructions, from any given current state of the machine, in one instruction, it is impossible to reach most others. I.e. the $2^N \times 2^N$ mapping $(P(x,y))$ from state before executing an instruction to state afterwards, is sparse. $P(x,y)$ is the Markov transition matrix. However each instruction must put the machine into a new state. So at least one element in each column of $P(x,y)$ will be non zero. With I instructions, the new state could be upto I different states. That is upto I elements in each column of $P(x,y)$ can be non zero. If each instruction is equally likely, $P(x,y) = 0$ or $1/I \ldots$ Since values of $2/I$ or more correspond to two or more different instructions changing the computers memory to identical patterns higher values of $P(x,y)$ are rare.

Since every column of P has 2^N elements, for a single instruction, the chance of any given $P(x,y)$ being zero is $(1-2^{-N})$. Considering all I independent instructions gives the chance of any given $P(x,y)$ being zero is $(1-2^{-N})^I$. Consider two instructions chosen at random. $P^2(x,y) = 0$, or $1/I^2$ or $\ldots$ or $2I/I^2$. The chance of any given element of $P^2(x,y)$ being zero is $(1-2^{-N})^{2I}$.

For a instructions, each element of $P^a(x,y)$ will be a multiple i (possibly zero) of I^{-a}. The values of i will be randomly distributed and follow a binomial distribution with $p = 1/2^N$, $q = 1-p$ and number of trials $= I^a$. So the mean of the distribution of i is $I^a/2^N$ and its standard deviation is $\sqrt{I^a \times 1/2^N \times (1-1/2^N)}$. For large I^a the distribution will approximate a Normal distribution. If $I^a \gg 2^N$, even for large 2^N, practically all i will lie within a few (say 5) standard deviations of the mean. I.e. the smallest value of i in any column will be more than $I^a/2^N - 5\sqrt{I^a \times 1/2^N}$. This allows us to use to minorization [2] to put an upper bound on the difference between the actual distribution of states μ_{ka} after ka randomly chosen instructions and the limiting distribution pi after many random instructions $||\mu_{ka} - \pi|| \le (1-\beta)^k$. Where β is the sum of the minimum values of the entries in each column of the matrix P^a. I.e. $\beta = \sum_j \min_i P_{ij}$. So $\beta \ge (1 - 5\sqrt{I^{-a} \times 2^N})$.

Let $a = \frac{-2\log(\alpha/5) + N\log 2}{\log I}$, and so $\beta \ge 0.5$. $||\mu_l - \pi|| \le (1-\beta)^{\lfloor l/a \rfloor}$ Choosing a target value of $||\mu_l - \pi||$ of 10% gives:

$$l \quad \le \quad \frac{15.3 + 2.30\ N}{\log I} \tag{1}$$

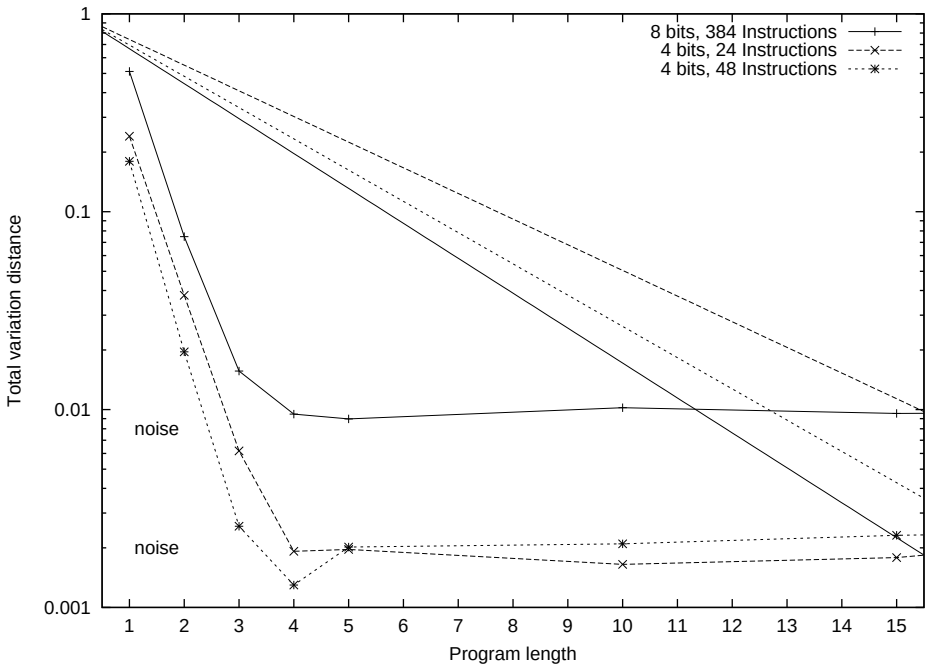

Figure 1: Convergence of three random machines, measured with a million random functions at each length. Note allowing for noise, measurements are within theoretical upper bound, Inequality (1) (straight lines). Also note large instruction sets are needed to ensure randomly connected computers can access all their memory states.

Note this predicts quite convergence for our randomly wired computer. Inequality (1) bounds the length of random programs need to be to ensure, starting from any state, the whole computer gets close to its limiting distribution. If we only consider the output register (of m bits) Inequality (1) becomes $l \leq (15.3 + 2.30\ m)/\log I$, cf. Figure 1.

For brevity we have not included the details whereby this approach is extended to cover functions implemented by the average computer, however the results were sumarised in abstract.

2 Discussion

The model does not cover programs that contain instructions that are executed more than once. I.e. no loops or jumps. This is, of course, a big restriction. However, many problems have been solved by GP systems without such loops or recursive function calls. The difficulty for the *proofs* is that, in general, repeating (a sequence of) random instructions need not give, on average, the same results as the same number of random instructions chosen independently. (If the loop contains enough random instructions to reach the limiting distribution then the problem does not arise because the input to the next iteration to the loop is already in the limiting distribution and so will remain there.) Similarly, there is no problem if the loop is followed by a large number of random instructions.

References

[1] W. B. Langdon. How many good programs are there? How long are they? In K. A. De Jong *et al.* eds, *Foundations of Genetic Algorithms VII*, pp 183–202, Morgan Kaufmann. 2003.

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